Sistemas Informáticos
Curso 2003-2004

UTC: Herramienta esteganalítica modular

Ajay Arjandas Daryanani
Raquel Fraile Alonso
Aitor Pérez Cano

Dirigido por:
Prof. Luis Javier García Villalba
Dpto. Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

INDICE DE CONTENIDOS

DESCRIPCIÓN DEL CUADERNO

Este documento trata de explicar el motivo de la creación de la aplicación UTC(Under The Carpet), cuyo nombre se debe al simil de buscar algo que aparentemente no existe.Esta aplicación se basa en un proyecto fin de carrera que está basado en el arte de la Esteganografía que se trata de una técnica cuya finalidad es ocultar datos detrás de otros datos , de manera que no sea visible a primera vista,al menos.La Esteganografía es una ciencia muy difundida pero no tan estudiada como sería deseable.Se realiza este tipo de técnica desde hace muchos años pero al existir tantos tipos de programas diferentes que tienen como finalidad la ocultación de información de manera digital,es difícil conocer todas las maneras posibles.

Con UTC intentamos englobar las metodologías más usadas hasta ahora en una sola aplicación en forma de módulos siendo cada uno de ellos independiente al resto, de manera que en el futuro se puedan añadir más métodos nuevos de Esteganografía que no se hayan implementado o que aparezcan nuevos en la red.Por lo tanto, UTC no busca encontrar nuevos métodos esteganográficos ,sino englobar en una sola aplicación los programas que ya existen y todo eso lo hemos implementado en Borland C++.

En la primera sección vamos a explicar por qué consideramos útil esta aplicación y los conceptos esteganográficos en los que está basada y la no existencia hasta ahora de una aplicación que use diferentes métodos esteganográficos para buscar información en oculta en la red.

En la segunda sección explicaremos cuáles son cada uno de los módulos y por qué los hemos considerado importantes para incluirlos en UTC.No buscamos realizar una documentación exhaustiva de cada uno de ellos sino explicar su funcionamiento y la forma de detección de cada uno de ellos, que en definitiva es en lo que se basa UTC.No hemos implentado la forma de recuperar dichos datos,sólo la búsqueda de información oculta,la evidencia de qué hay datos ocultos por detrás de ciertos ficheros, sin importarnos qué datos son.Por supuesto esto podría incluirse como un nuevo submódulo en cada uno de los módulos implementados.

SECCIÓN I: INTRODUCCIÓN

Lista de Contenidos

1. Conceptos sobre Esteganografía

La Esteganografía es el arte de esconder datos detrás de otros datos sin que la persona que mira el fichero sea capaz de apreciar la diferencia.

La palabra Esteganografía viene del griego y significa "el arte de ocultar cosas",en un principio la esteganografía lo único que busca es detectar la presencia de datos ocultos pero no la recuperacion o la conversión de dichos datos, sino que sólo le preocupa el conocimiento de que esos datos existen y poder diferenciar los ficheros que tienen mensajes ocultos y los que no los tienen.

Existen muy distintos softwares en Internet que se encargan de detectar estegos en la red , el funcionamiento de este tipo de software es muy diferente según la forma de buscar la información que tenga.La idea de "Under The Carpet" surgió porque después de estudiar y buscar en internet las distintas técnicas de Esteganografía que existían no encontramos ninguna aplicación que englobara a todas ellas en una sola, de manera que se pudiera trabajar tan sólo insertanto ficheros bajados de la red con distintas extensiones y se pudiera detectar cuales contienen imágenes ocultas y cuáles no basándonos en varias técnicas de programación y encriptación.

Por supuesto somos conscientes de lo difícil qué es tener acceso a todos los métodos de esteganografía que existen y por eso hemos implementado en forma de módulos para que el desarrollo de cada uno de los softwares sea independiente y se puedan ir insertando en el futuro nuevos módulos que vayan apareciendo u otros que no hayamos implementando pero que existan en la esteganografía y se estén utilizando.

La Esteganografía se encarga de ocultar información detrás de ficheros aparentemente normales para el usuario en muy distintos tipos de ficheros:JPG,WAV,MP3,BMP...Los distintos métodos que se usan son de muy distintos tipos y según la complejidad de desarrollo que tengan así de fácil es la detección de información oculta.Algunos de ellos son tan simples que ni siquiera se encargan de encriptar los datos , por si estos son detectados sino que tan sólo los añaden al final del fichero , de manera que no son visibles a primera vista pero su detección es extremadamente sencilla,lo que les convierte en programas muy débiles a posibles ataques.
Es el caso de los softwares que añaden información al final de los ficheros JPG.0

2. Especificación

2.1 El reto del profesor

La idea principal tanto del profesor como del grupo era modularizar lo máximo posible la aplicación de manera que el resultado final fuera algo completamente clasificado por tipos de programa en donde fuera posible eliminar y añadir nuevos módulos sin necesidad de tocar el núcleo del código y de forma que cada uno clase sea independiente del resto, ya que un proyecto como UTC que fuera cerrado y no permitiera el acceso modularizable al mismo no tendría mucho sentido, ya que la Esteganografía se trata de una técnica que avanza con el tiempo, apareciendo nuevos métodos de ocultamiento de información.

2.2 Propósito “Under The Carpet”

Esteganografía no está demasiado explotada en el mundo de la investigación,o al menos no tanto como se debería, teniendo en cuenta que se cree que día tras día se producen envios de ficheros con información oculta en internet.Por eso nosotros pensamos en este proyecto, no como investigación de nuevos métodos sino como implementación de los más usados en una sola apliación que permita analizar un número grande de ficheros de diferentes extensiones bajadas de internet con diferentes técnicas de una sola vez sin necesidad de analizarlos con cada uno de

los programas, para los que algunos de ellos es necesario pagar para poder hacer uso de ellos.

Se trata de un software que se colgará de la red en la página **www.sourceforge.net** donde se podrán seguir añadiendo módulos de detección esteganográfica que vayan apareciendo y nuevas funcionalidades entre ellas:nuevas extensiones de ficheros, nuevos formatos(audio,mp3..) que también se usa mucho en esteganografía,recuperación de los datos ocultos detectados...

2.3. Metodología del trabajo del grupo

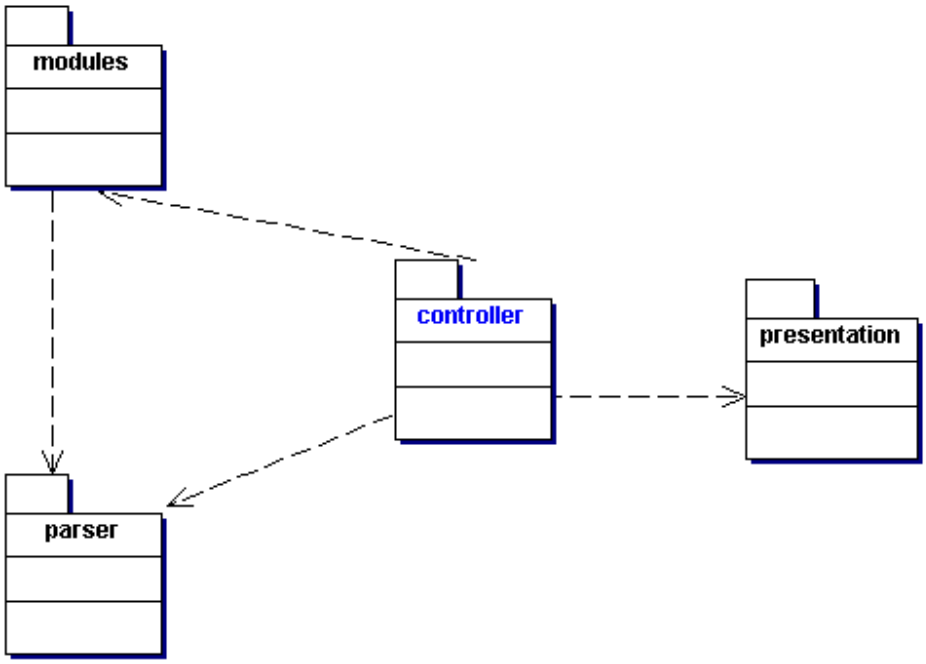
El grupo está formado por tres participantes y no hemos llevado a cabo una metodología conocida pero el desarrollo del trabajo se ha llevado a cabo con mucha comunicación informal entre los integrantes del grupo y dividiéndose el trabajo , en primer lugar en investigación de diferentes técnicas encontradas,poniendo en común toda la información encontrada y después dividiendo el trabajo por módulos desarrollando cada uno de los integrantes módulos que usaban clases comunes pero que son completamente independientes del resto.Hemos llevado a cabo reuniones semanales donde hemos puesto en consenso como iba el trabajo,las posibles dificultades encontradas y encontrando las posibles soluciones entre el grupo.

3. Diseño

3.1 Objetivos de Diseño

En la especificación, un requisito de especial importancia era la **extensibilidad** y la **reutilización**. Basándonos en esos criterios, decidimos dividir la arquitectura del sistema en varias entidades que fuesen lo más independientes posibles:

1. **Controlador** general del proceso de detección. Él se encargará de saber qué tareas realizar para detectar contenidos ocultos en un medio específico.
2. **Lector** de ficheros y empaquetador de estos datos leídos en los “medios” correspondientes. Esto es, si leíamos un fichero .jpg, debíamos proveer una manera de acceder a sus propiedades: sus bytes, coeficientes CFT, etc.
3. **Módulos** que detectan programas de ocultación de contenidos. Para cada programa de esteganografía, habrá un módulo que lo detecte. Pero también puede haber módulos que implementen un método general de detección, como los “ataques visuales”, “RS attack”, “PoV attack”, etc.
4. **GUI** para mostrar los resultados, errores o cualquier mensaje por pantalla.

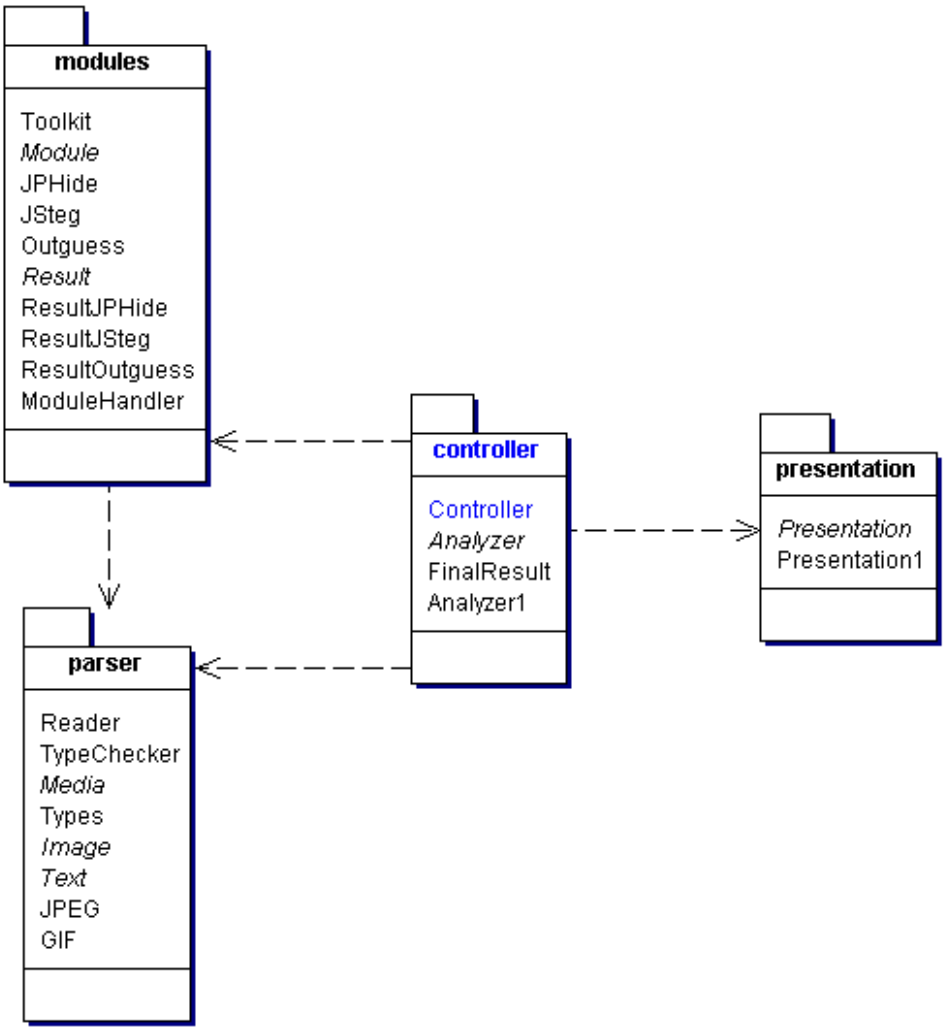


Como puede verse, no es más que la aplicación del *patrón MVC* (Model, View, Controller): El modelo de negocio se reparten entre Parser y Modules; el controlador es Controller, y la vista está a cargo de Presentation.
Las entidades del diagrama están organizadas alrededor del núcleo de UTC: el Controlador. Quien quiera ampliar las capacidades de UTC, tan sólo debe fijarse en el paquete de su interés y en el Controlador, pues éste es el que relaciona a unas entidades con otras: reducimos las relaciones de "muchos a muchos" a "muchos a uno". Las entidades no se comunican entre sí, si no es a través del Controlador. Luego veremos que, dada la estrecha relación que hay entre el proceso de detección (los Módulos) y el medio sobre el que efectúan dicho proceso (Medios que empaqueta el Lector), también habrá interacción entre ellos.

Además, lo único que los "clientes" de UTC deben/necesitan conocer, es la clase que implemente la funcionalidad del Controlador, puesto que a través de ella se proporcionan los métodos necesarios para llevar a cabo toda la detección. Así, cualquiera que quiera usar nuestro código, sólo debe fijarse en la interfaz del Controlador, lo que simplifica mucho la facilidad de uso de nuestro software.

3.2 Primera Aproximación

Una vez la estructura general estaba clara, sólo había que distribuir la funcionalidad en clases dentro de cada una de las entidades.

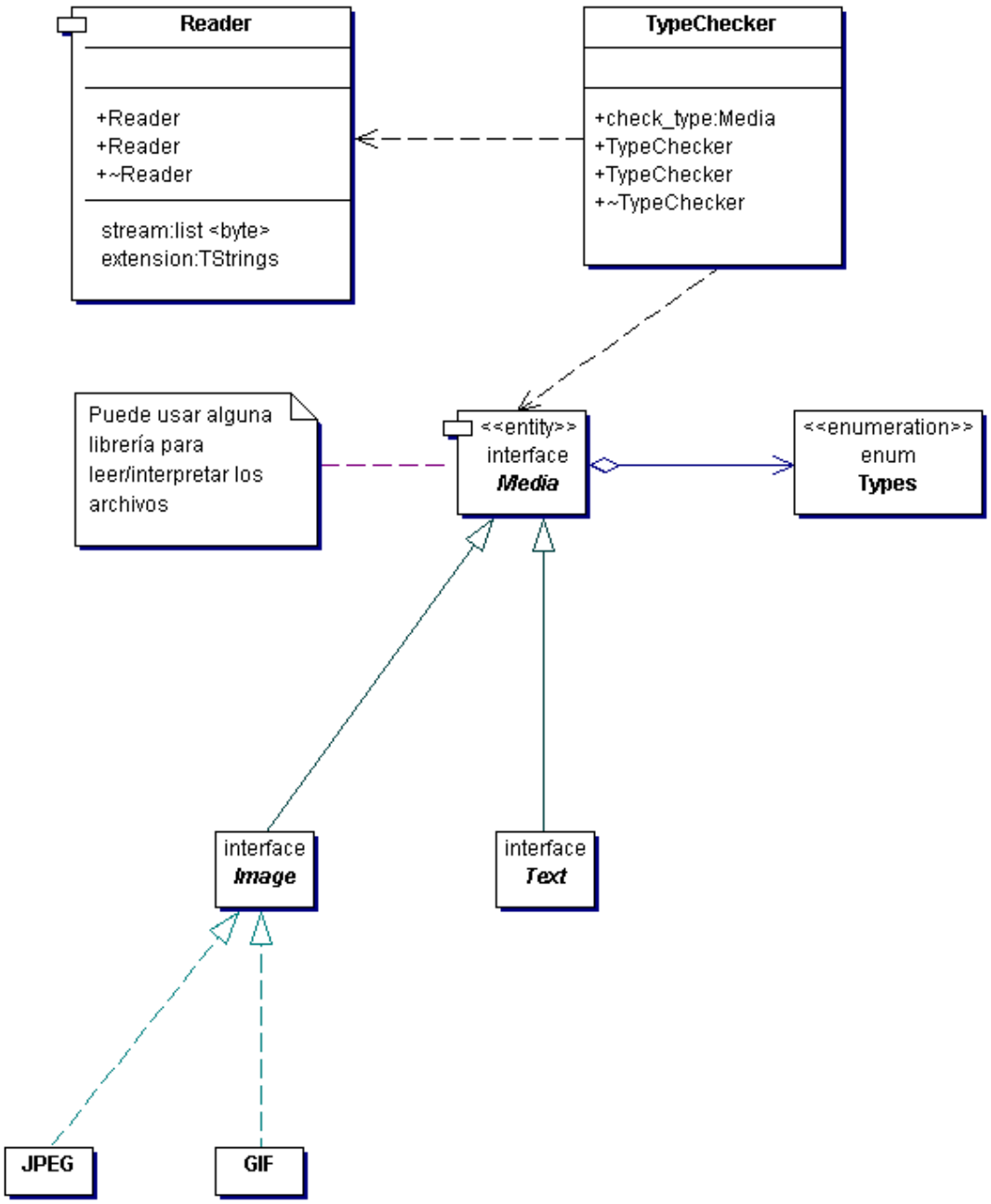


Parser

Lee el fichero y lo cataloga en el Medio adecuado.

- Reader: lee el fichero en un FILE.
- TypeChecker: clasifica el fichero, creando el medio adecuado.
- Media: interfaz de todos los medios. Usa alguna biblioteca para interpretar el formato.
    - Image: interfaz para las imágenes.
        - JPEG: tiene los métodos para acceder a los coeficientes DCT, hacer histogramas…
        - GIF: métodos para acceder a la paleta de colores y al raster.
    - Text: interfaz para textos.

- Types: enumeración para controlar todos los tipos de medios que el programa soporta.

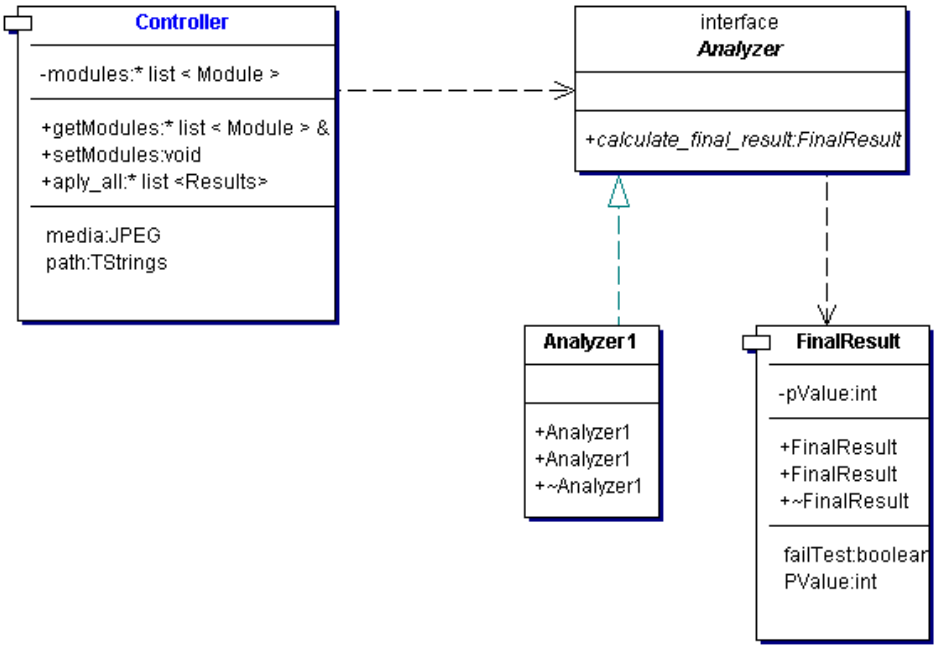


Controller

Dirige la ejecución del programa.

- Controller: maneja el proceso de detección.
- Analyzer: interfaz para analizar los resultados parciales de cada módulo ejecutado.

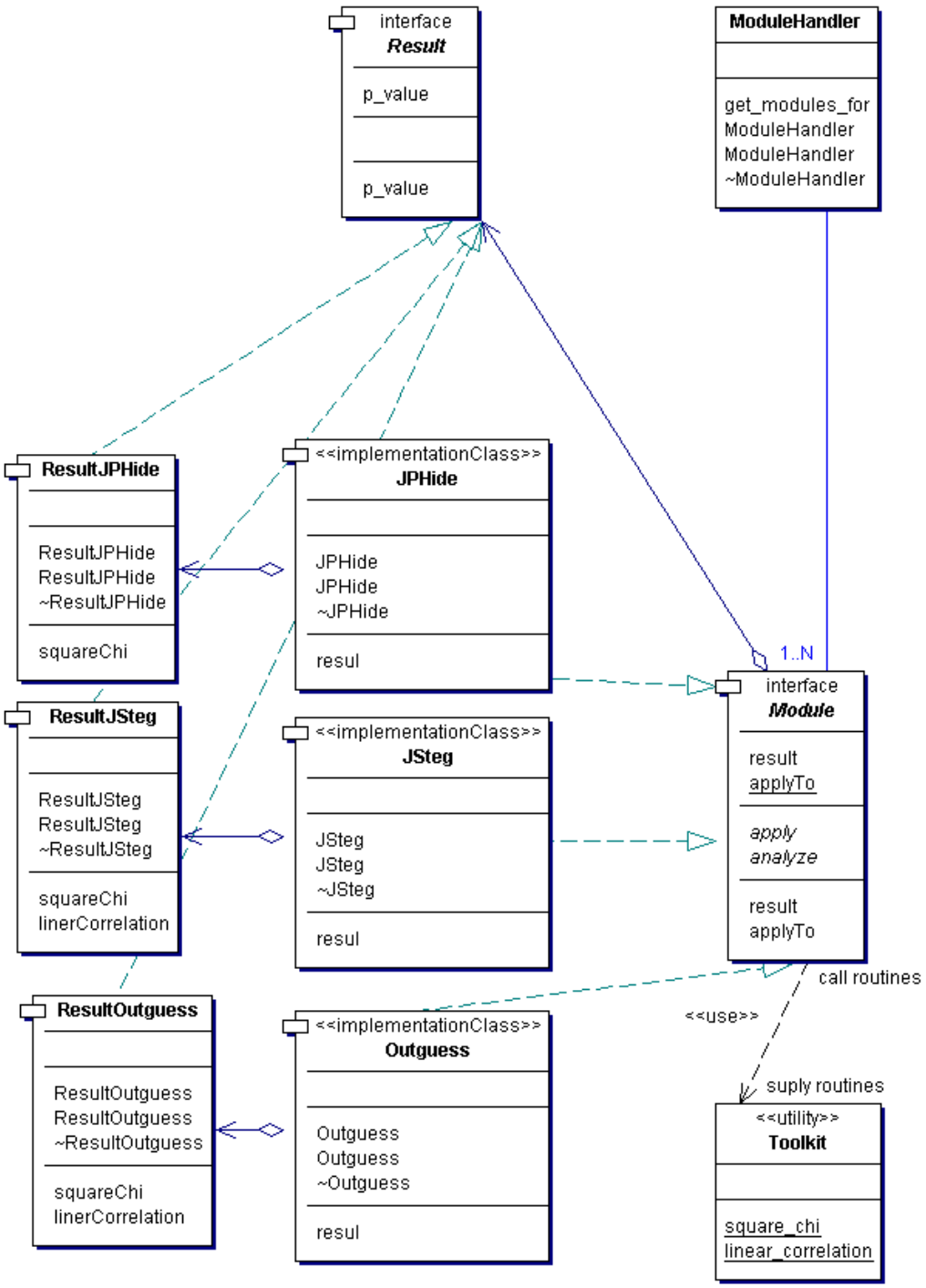
  - DefaultAnalyzer: pondera los resultados parciales para conseguir un valor final.
- FinalResult: resultado final tras el análisis.



Modules

Detecta los diferentes programas de esteganografía.

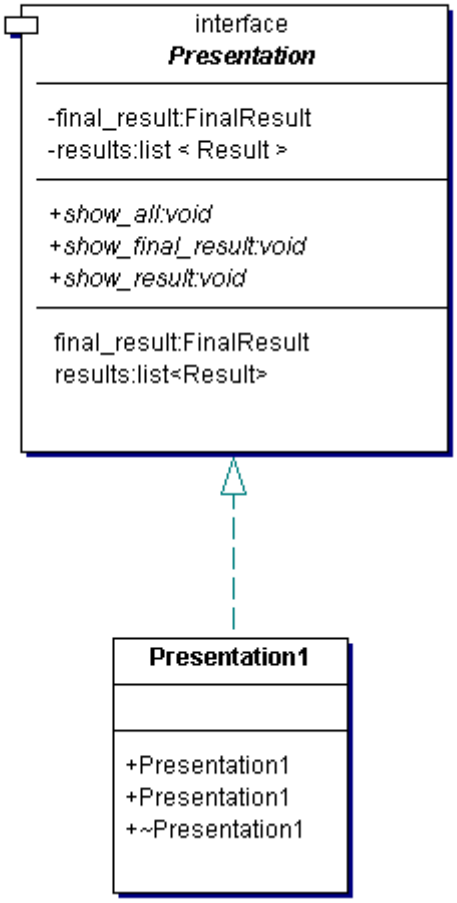
- ModuleHandler: conoce todos los módulos y para qué medios están disponibles.
- Module: interfaz común a todos los módulos.
  - JPHide: detecta JPHide (random LSB).
  - JSteg: detecta JSteg (secuential LSB).
  - Outguess: detecta Outguess (spread LSB with statistical compensation).
- Result: interfaz común para almacenar los resultados de cada módulo.
  - ResultJPHide: almacena los resultados de JPHide.
  - ResultJSteg: almacena los resultados de JSteg.
  - ResultOutguess: almacena los resultados de Outguess.
- Toolkit: Repositorio de funciones estadísticas y utilidades (test_chi_cuadrado, transformación de coeficientes CFT, etc.). Todos sus métodos son estáticos, por lo que no es necesario crear una instancia.



Presentation

GUI para la entrada/salida de datos con el usuario.

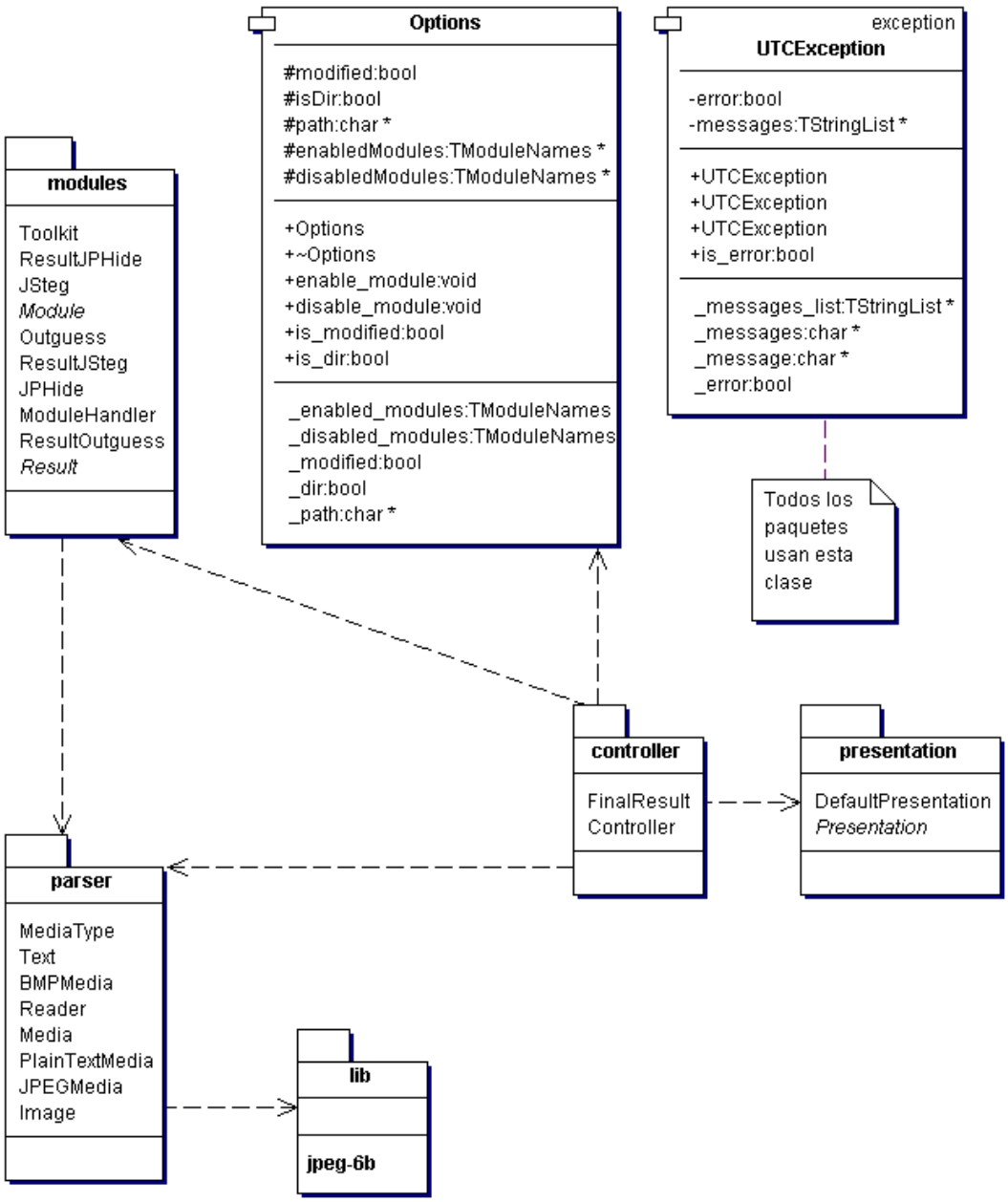
- Presentation: interfaz para presentar los resultados.
- Presentation1: presenta los resultados por la salida estándar (modo consola).



3.3 Versión 0.1

En el momento de alcanzar la versión 0.1 ya teníamos una idea muy aproximada de cómo sería el diseño definitivo. Aparecieron muchos métodos nuevos, algunas clases cambiaron de nombre, aumentaron o disminuyeron sus responsabilidades, y desaparecieron algunas otras. Las 2 clases siguientes, no están dentro de ningún subpaquete:

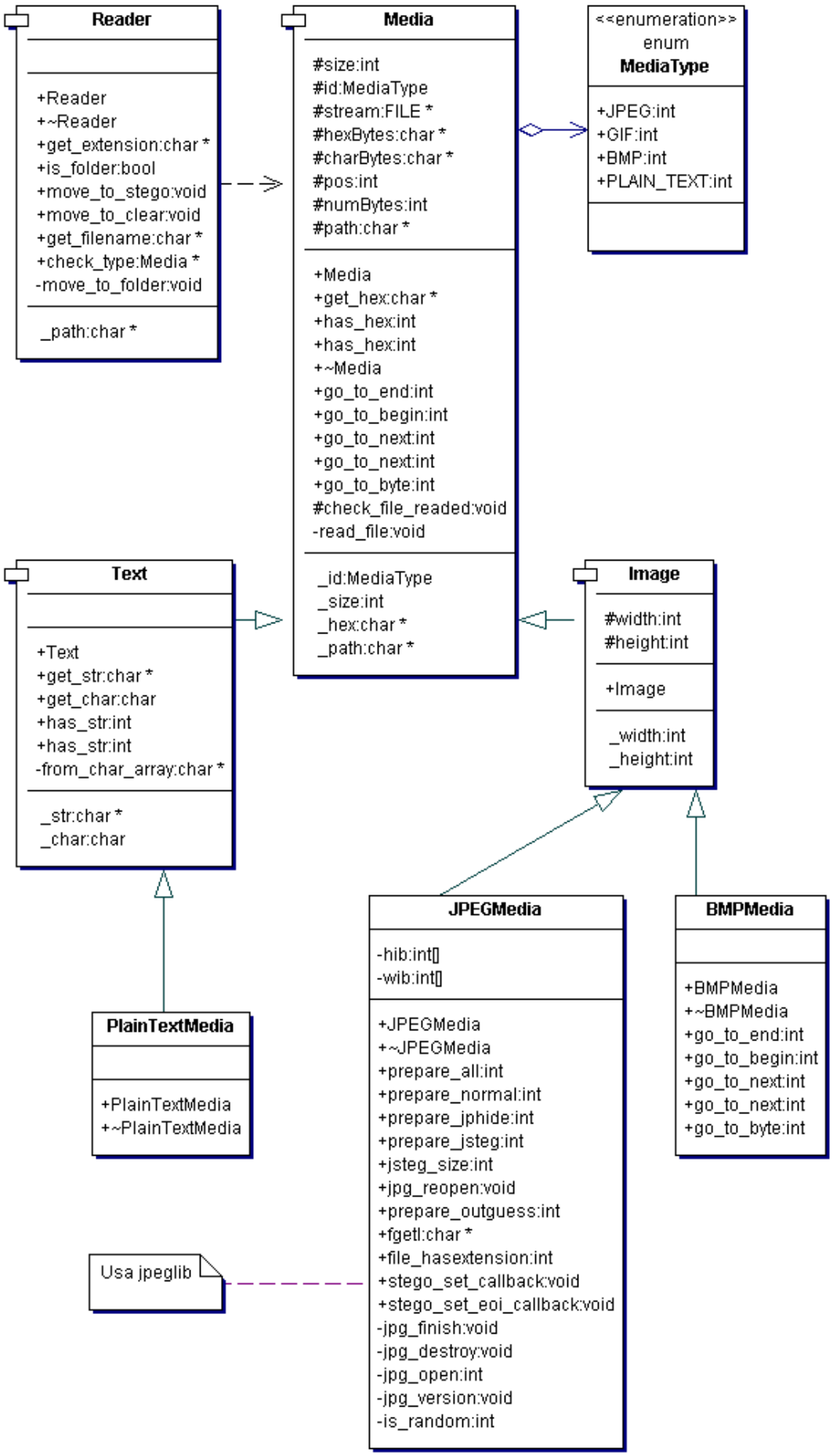
- UTCException es la clase de excepciones personalizada. La utilizan todas las demás clases, para indicar que se ha producido un error o un aviso.
- Options aglutina las opciones que le hemos pasado al programa en su inicio (en este caso, a través los “args” del “main”). Así tenemos una interfaz limpia entre el Controller y las distintas y cambiantes opciones iniciales.



Parser

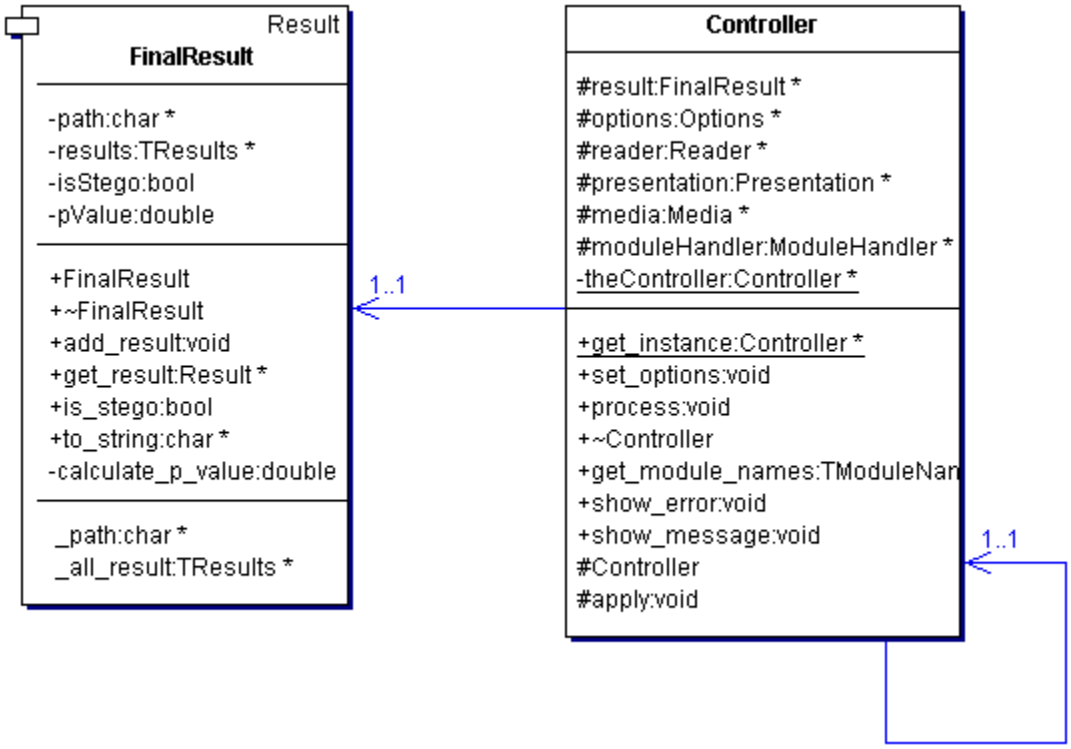
- TypeChecker y Reader se fundieron en una sola clase: En principio no sabíamos si el método para distinguir un tipo de medio de otro iba a ser más o menos elaborado (podríamos haber comprobado que el formato del archivo es realmente el que aparenta ser con la extensión del nombre del fichero). Pero al final nos decantamos por tener en cuenta sólamente la extensión, por lo que con una distinción de casos nos podría valer. Así, en Reader usamos el *patrón Factory*: a partir de un parámetro (la ruta) devolvemos la instancia adecuada de Media.

- Al añadir módulos que detectaban otros tipos de medios, como BMP y texto claro, tuvimos que añadir nuevas subclases de Media. Desechamos GIF, pues los programas de esteganografía que pensábamos manejar no utilizan este formato.


Reader
+Reader
+~Reader
+get_extension:char *
+is_folder:bool
+move_to_stego:void
+move_to_clear:void
+get_filename:char *
+check_type:Media *
-move_to_folder:void
_path:char *
Media
#size:int
#id:MediaType
#stream:FILE *
#hexBytes:char *
#charBytes:char *
#pos:int
#numBytes:int
#path:char *
+Media
+get_hex:char *
+has_hex:int
+has_hex:int
+~Media
+go_to_end:int
+go_to_begin:int
+go_to_next:int
+go_to_next:int
+go_to_byte:int
#check_file_readed:void
-read_file:void
_id:MediaType
_size:int
_hex:char *
_path:char *
<<enumeration>>
enum
MediaType
+JPEG:int
+GIF:int
+BMP:int
+PLAIN_TEXT:int
Text
+Text
+get_str:char *
+get_char:char
+has_str:int
+has_str:int
-from_char_array:char *
_str:char *
_char:char
Image
#width:int
#height:int
+Image
_width:int
_height:int
PlainTextMedia
+PlainTextMedia
+~PlainTextMedia
JPEGMedia
-hib:int[]
-wib:int[]
+JPEGMedia
+~JPEGMedia
+prepare_all:int
+prepare_normal:int
+prepare_jphide:int
+prepare_jsteg:int
+jsteg_size:int
+jpg_reopen:void
+prepare_outguess:int
+fgetl:char *
+file_hasextension:int
+stego_set_callback:void
+stego_set_eoi_callback:void
-jpg_finish:void
-jpg_destroy:void
-jpg_open:int
-jpg_version:void
-is_random:int
BMPMedia
+BMPMedia
+~BMPMedia
+go_to_end:int
+go_to_begin:int
+go_to_next:int
+go_to_next:int
+go_to_byte:int
Usa jpeglib

Controller

- Las clases FinalResult y Analyzer pasaron a ser una sola: Pensamos que habría que hacer algún tipo de ponderación o transformación de los resultados parciales de los módulos para llegar al resultado final, pero la función que elegimos fue el máximo p-valor de todos los resultados, por lo que aunamos esta característica en la clase FinalResult.
- En Controller utilizamos el *patrón Singleton*, que permite controlar el número de instancias de una clase que se pueden tener. Dado que sólo tiene sentido tener un único controlador, nos vino bien usar esta alternativa.

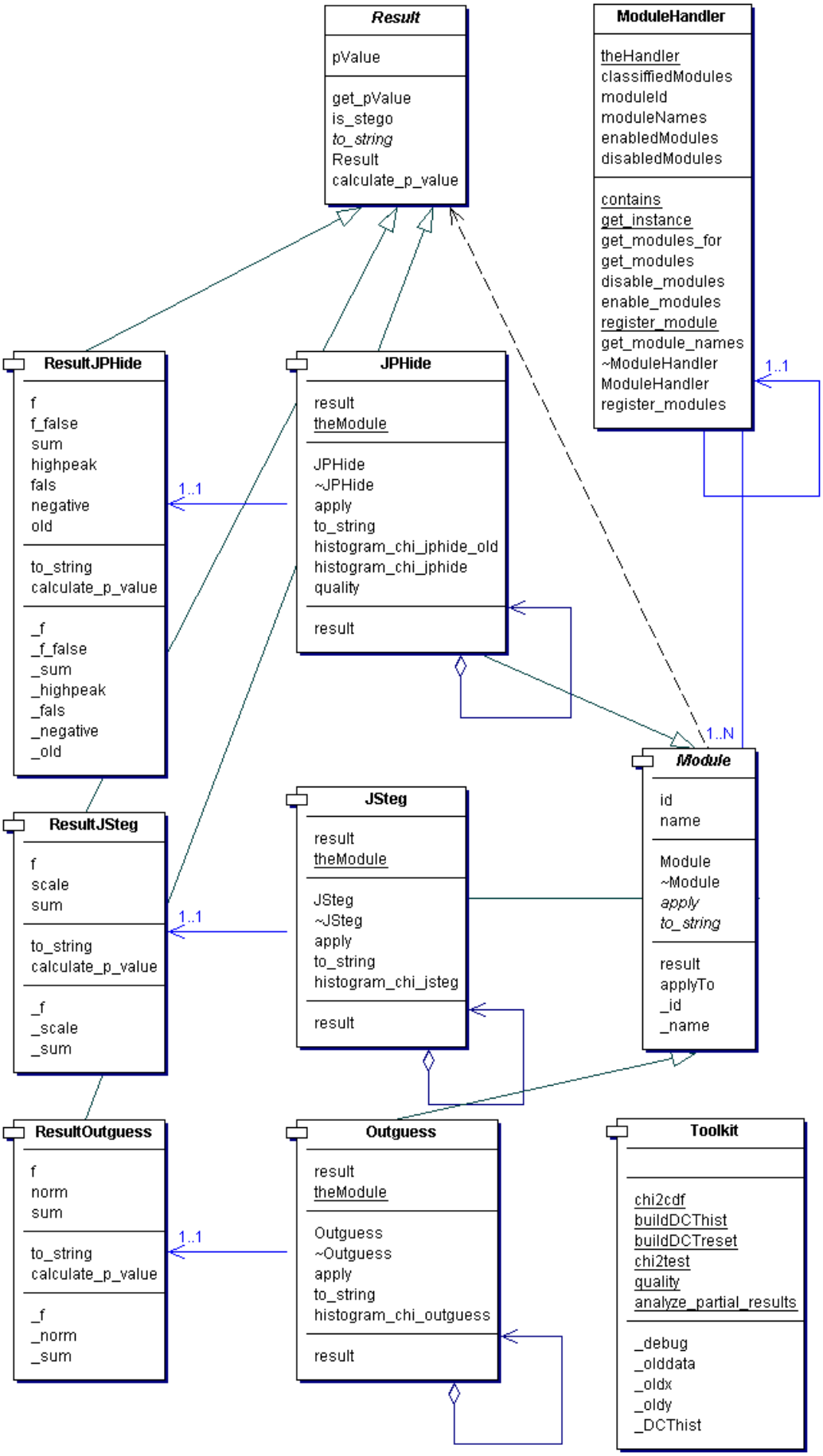


Modules

- En ModuleHandler también usamos el *patrón Singleton*, pues va a ser la única instancia que maneje a todos los módulos.
- En el resto de los módulos, tenemos un miembro estático del mismo tipo que la clase que se está definiendo. Es una variación del *patrón Mediator*, ya que en vez de pasar una instancia del ModuleHandler a los Modules, usamos un método estático de ModuleHandler para registrar a los Modules. Su utilidad reside en que así logramos que el ModuleHandler conozca todos los módulos que existen. Dado que lo primero que se ejecuta son las inicializaciones de los miembros estáticos, logramos tener antes que nada todos los módulos disponibles. Así nos evitamos tener un tipo enumerado (como MediaTypes) que hay que actualizar cada vez que se añade un nuevo módulo. El anterior tipo enumerado tiene sentido, ya que no se

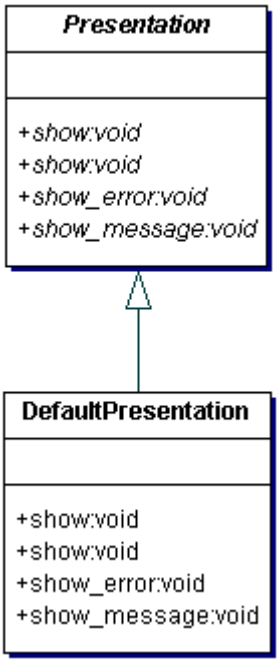
va a añadir al mismo ritmo el número de medios distintos que el de nuevos módulos.

En el diagrama sólo se muestran los mismos módulos que en la primera aproximación. Si estuviesen todos no se vería la estructura claramente, que, por otro lado, es simétrica a la de los módulos que sí aparecen:


Result
pValue
get_pValue
is_stego
to_string
Result
calculate_p_value
ModuleHandler
theHandler
classifiedModules
moduleId
moduleNames
enabledModules
disabledModules
contains
get_instance
get_modules_for
get_modules
disable_modules
enable_modules
register_module
get_module_names
~ModuleHandler
ModuleHandler
register_modules
1..1
ResultJPHide
f
f_false
sum
highpeak
fals
negative
old
to_string
calculate_p_value
_f
_f_false
_sum
_highpeak
_fals
_negative
_old
JPHide
result
theModule
JPHide
~JPHide
apply
to_string
histogram_chi_jphide_old
histogram_chi_jphide
quality
result
1..1
1..N
Module
id
name
Module
~Module
apply
to_string
result
applyTo
_id
_name
ResultJSteg
f
scale
sum
to_string
calculate_p_value
_f
_scale
_sum
JSteg
result
theModule
JSteg
~JSteg
apply
to_string
histogram_chi_jsteg
result
1..1
ResultOutguess
f
norm
sum
to_string
calculate_p_value
_f
_norm
_sum
Outguess
result
theModule
Outguess
~Outguess
apply
to_string
histogram_chi_outguess
result
1..1
Toolkit
chi2cdf
buildDCThist
buildDCTreset
chi2test
quality
analyze_partial_results
_debug
_olddata
_oldx
_oldy
_DCThist

Presentation

No cambia mucho, salvo la clase concreta, que pasa a llamarse DefaultPresentation. Ya no tiene atributos de instancia, pues no es necesario almacenarlos para que realice su función (se le pasan por parámetro).

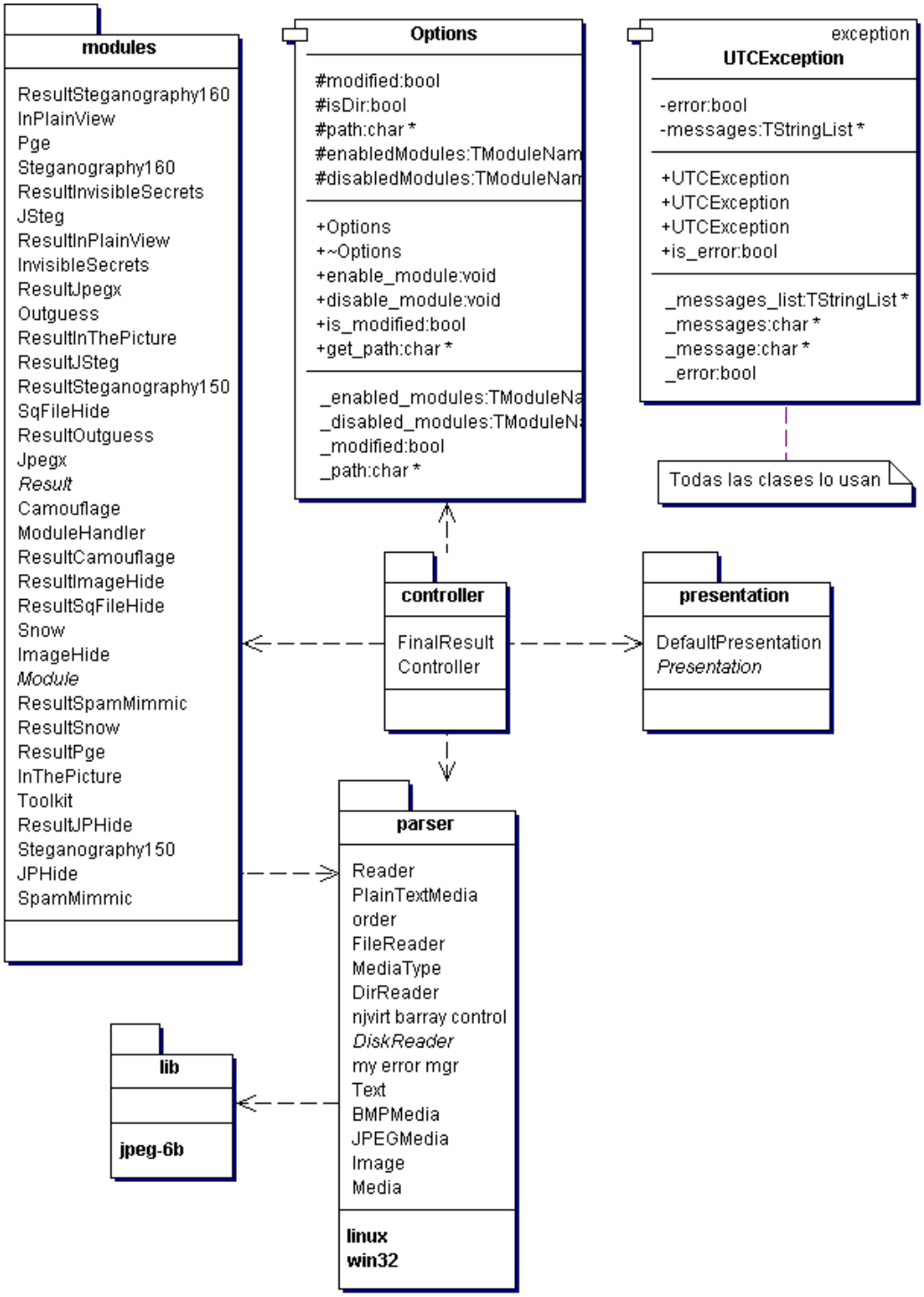


Lib

En un comienzo no sabíamos de qué librerías o herramientas nos íbamos a servir para decodificar los distintos formatos de ficheros (pensamos en ImageMagic). Pero nos dimos cuenta que con la veterana pero efectiva jpeglib teníamos resuelto el problema. Aún así, usamos una versión modificada por Provos, para guardar la compatibilidad con StegDetect. Para el resto de formatos, seríamos nosotros mismos los que extraeríamos los datos.

3.4 Versión 0.2

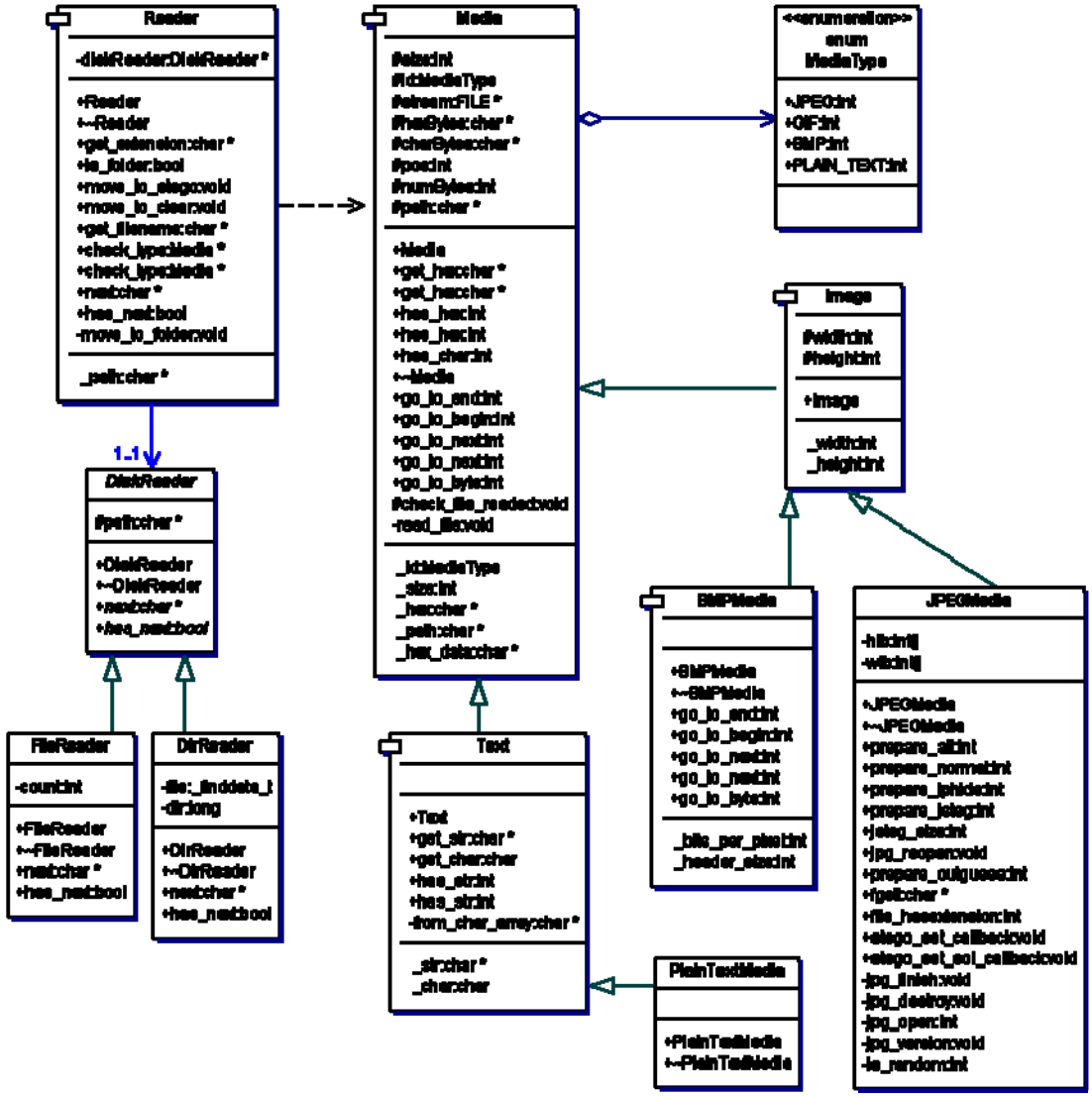
El diseño actual no varía casi nada respecto a la versión anterior, lo que nos puede dar una idea de que estamos cerca del definitivo. Tan sólo añadimos variaciones según se trate de la adaptación para Windows o para Linux.



Parser

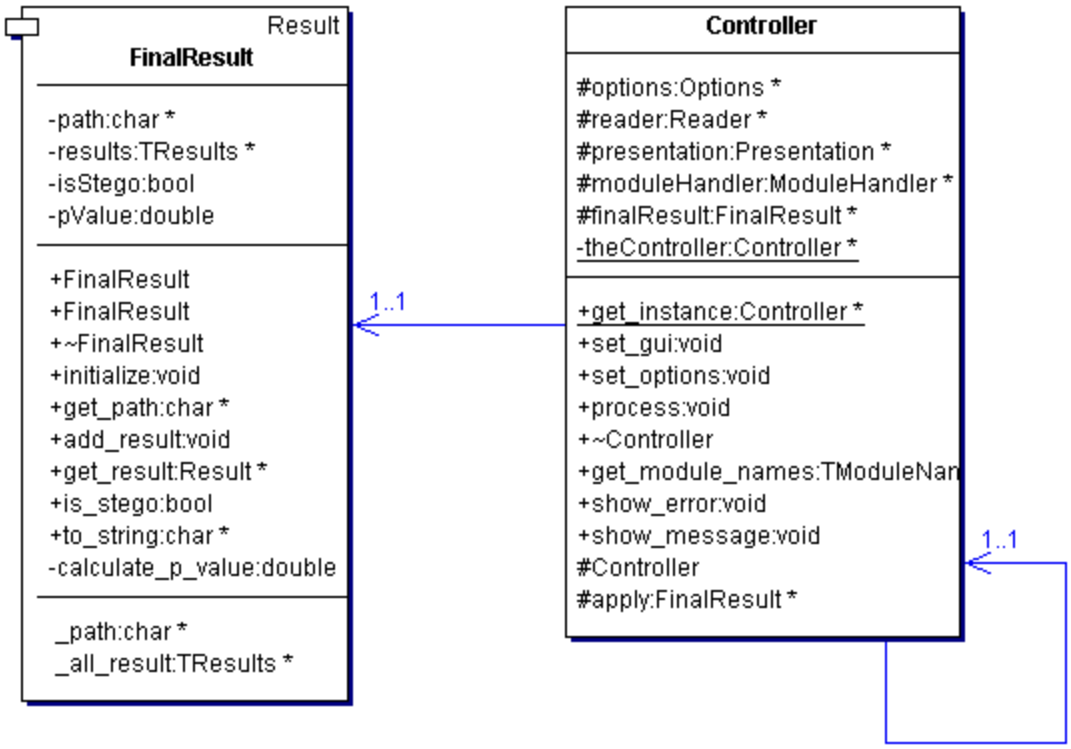
En esta iteración añadimos la posibilidad de leer directorios completos. Ya que el acceso a directorios es dependiente del sistema operativo, tuvimos que hacer 2 versiones: una para Linux y otra para WindowsNT/2000/XP.

En Reader usamos el *patrón Strategy*: dependiendo de si queremos leer un directorio o un fichero ordinario, usamos una instancia de FileReader o DirReader, que implementan la misma interfaz: DiskReader.



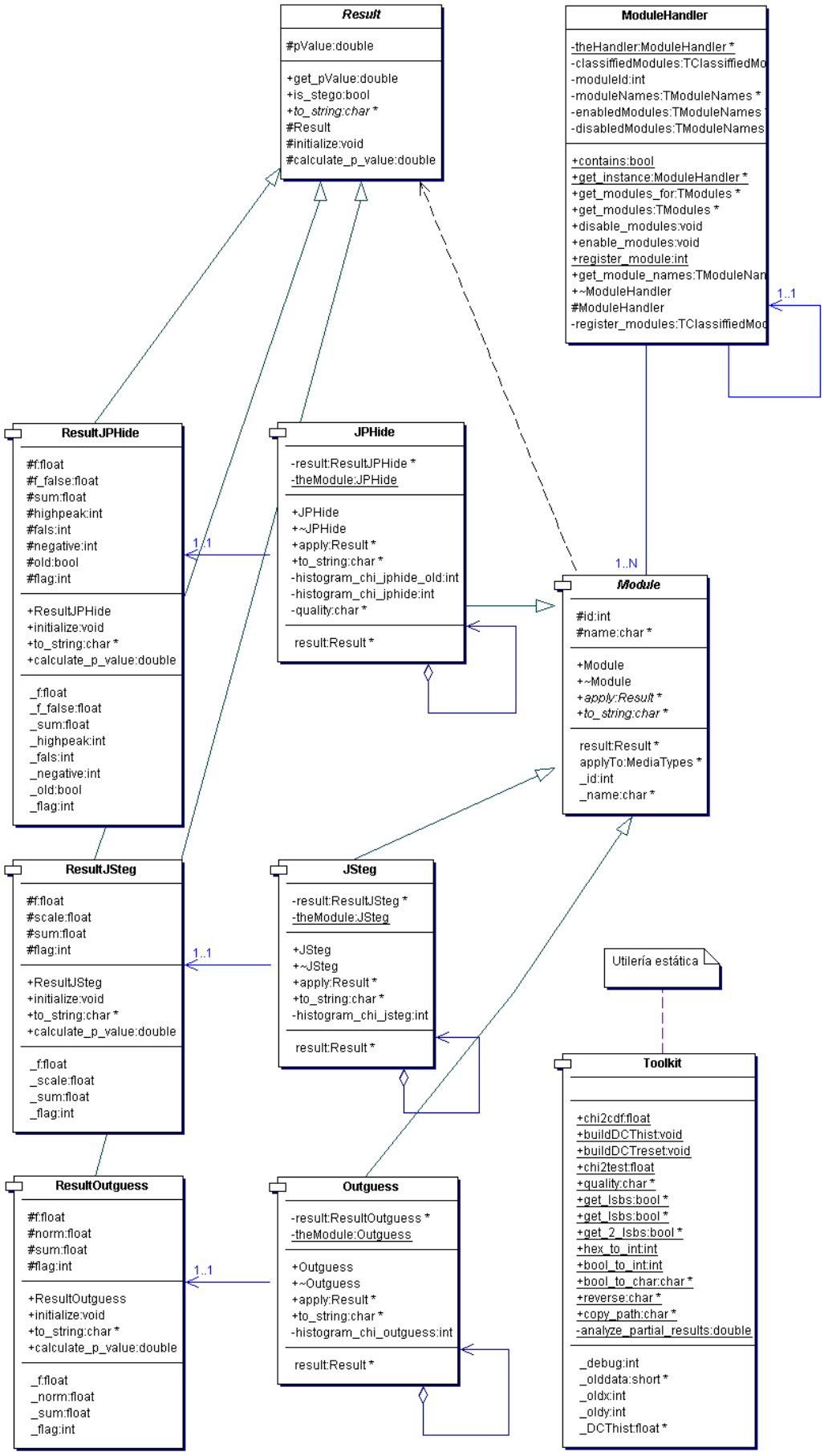
Controller

Aunque en este diagrama no aparece Options, es muy probable que en la próxima versión sí lo haga, puesto que está muy estrechamente relacionado con Controller. Usamos el *patrón Strategy* para indicar al Controller qué instancia en concreto que implemente Presentation debe usar para mostrar la GUI.
Tan sólo indicar que Media ya no es un atributo de instancia de Controller, pues Reader nos devolverá una nueva instancia de Media según sea el tipo de fichero que lea.



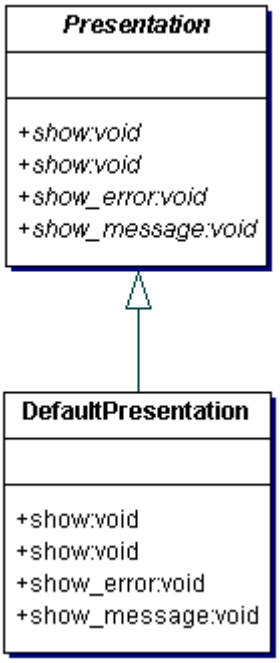
Modules

No ha cambiado nada, salvo que los Result tienen un nuevo método “initialize” para que no se vayan acumulando los resultados entre la detección de varios ficheros. Sigue aumentando el número de métodos de Toolkit, y también la cantidad de módulos, pero no se muestran en el diagrama porque no aportan nada nuevo al diseño y por economía de espacio.


Result
#pValue:double
+get_pValue:double
+is_stego:bool
+to_string:char *
#Result
#initialize:void
#calculate_p_value:double
ModuleHandler
-theHandler:ModuleHandler *
-moduleId:int
-moduleNames:TModuleNames *
+contains:bool
+get_instance:ModuleHandler *
+get_modules_for:TModules *
+get_modules:TModules *
+disable_modules:void
+enable_modules:void
+register_module:int
+~ModuleHandler
#ModuleHandler
1..1
ResultJPHide
#f:float
#f_false:float
#sum:float
#highpeak:int
#fals:int
#negative:int
#old:bool
#flag:int
+ResultJPHide
+initialize:void
+to_string:char *
+calculate_p_value:double
_f:float
_f_false:float
_sum:float
_highpeak:int
_fals:int
_negative:int
_old:bool
_flag:int
JPHide
-result:ResultJPHide *
-theModule:JPHide
+JPHide
+~JPHide
+apply:Result *
+to_string:char *
-histogram_chi_jphide_old:int
-histogram_chi_jphide:int
-quality:char *
result:Result *
1..1
1..N
Module
#id:int
#name:char *
+Module
+~Module
+apply:Result *
+to_string:char *
result:Result *
applyTo:MediaTypes *
_id:int
_name:char *
ResultJSteg
#f:float
#scale:float
#sum:float
#flag:int
+ResultJSteg
+initialize:void
+to_string:char *
+calculate_p_value:double
_f:float
_scale:float
_sum:float
_flag:int
JSteg
-result:ResultJSteg *
-theModule:JSteg
+JSteg
+~JSteg
+apply:Result *
+to_string:char *
-histogram_chi_jsteg:int
result:Result *
1..1
Utilería estática
Toolkit
+chi2cdf:float
+buildDCThist:void
+buildDCTreset:void
+chi2test:float
+quality:char *
+get_lsbs:bool *
+get_lsbs:bool *
+get_2_lsbs:bool *
+hex_to_int:int
+bool_to_int:int
+bool_to_char:char *
+reverse:char *
+copy_path:char *
-analyze_partial_results:double
_debug:int
_olddata:short *
_oldx:int
_oldy:int
_DCThist:float *
ResultOutguess
#f:float
#norm:float
#sum:float
#flag:int
+ResultOutguess
+initialize:void
+to_string:char *
+calculate_p_value:double
_f:float
_norm:float
_sum:float
_flag:int
Outguess
-result:ResultOutguess *
-theModule:Outguess
+Outguess
+~Outguess
+apply:Result *
+to_string:char *
-histogram_chi_outguess:int
result:Result *
1..1

Presentation

Todo igual que en la versión anterior. Comentar que como tratamos igual a todos los distintos resultados (nos basta con los atributos comunes que heredan de su superclase Result), no hace falta hacer nada especial. Si este no fuese el caso, se podría utilizar una variante del *patrón Visitor*: en vez de que cada módulo tuviese un método "show", o recibiese por parámetro a Presentation para ésta realizase el "show" para solventar este problema, se podría hacer un tratamiento individualizado de cada resultado en DefaultPresentation, según la subclase de Result que recibiésemos. Esto se lograría (en vez de distinguiendo casos: el número de Resultados, al igual que de módulos, es grande y variante) usando vinculación dinámica sobre los parámetros de una función "show". Por ejemplo: DefaultPresentation::show(ResultJPHide); DefaultPresentation::show(ResultJSteg); etc.



Lib

Como ahora nuestra aplicación es multiplataforma (Windows+Linux), tenemos dos versiones de nuestra jpeglib modificada.

4. Tipos de módulos UTC

En la siguiente sección hemos clasificado los módulos que hemos implementado según la forma de esconder la información.Los diferentes softwares esteganográficos se pueden dividir en diferentes categorías que van en función de su calidad:

1.Añadir datos al final del fichero visible(Camouflage,JpegX,SecurEngine para JPG,SqFileHide,Steganography...)

2.Insertar datos en algún campo comentario de la cabecera donde se indica la estrucutra del fichero(InvisibleSecrets para JPG,Steganozorus para JPG).

3.Embeber datos en el fichero visible en su "byte stream" de una manera fija que es lineal y secuencial.

4. Embeber datos en el fichero visible en su "byte stream" pero de una forma aleatoria dependiendo de una password (CryptArkan,BMPSecrets,Steganos para BMP,TheThirdEye,JPHide).

5. Embeber datos en el fichero visible en su "byte stream" pero de una forma aleatoria dependiendo de una password y cambiando otros bits del fichero que oculta la información para compensar las modificaciones inducidas por los datos ocultos y así evitar modificar propiedades estadísticas del fichero visible(Outguess,F5).

Realmente los programas clasificados en los puntos 1 y 2 son los que menos seguridad ofrecen y los más fáciles a la hora de detectar la presencia de estegos.Y los softwares más pontentes, como el Outguess o F5, ofrecen mucha más seguridad pero han conseguido ser ya detectados, aunque no es posible recuperar su datos porque están encriptados mediante algoritmos que no son en absoluto sencillos.

5. Historia de la aplicación

Para llegar a la versión actual (v0.2 Build 270), hemos recorrido un laborioso camino. Pero en nuestro ánimo está seguir con el desarrollo de nuestra herramienta más allá de donde está.

5.1 Prototipo: v0.1

El objetivo de esta versión era alcanzar un diseño estable. Para ver lo reutilizable y extensible que era nuestra arquitectura, usamos distintos tipos de medios e implementamos varios módulos de detección.
Sus principales características, relacionadas con la entidad que agrupa su funcionalidad son:

- Controller: Permite habilitar/deshabilitar módulos.
- Media: Maneja ficheros de imágenes (JPEG y BMP) y de texto (TXT).
- Modules: Detecta los programas Camouflage, InPlainView, JPHide, JSteg, Outguess, Snow y SpamMimmic. Además, en Toolkit proveemos los métodos chi2cdf; buildDCThist; buildDCTreset; chi2test; unify; quality; get_debug; get_olddata; get_oldx; get_oldy; y get_DCThist;
- Presentation: Permite presentar mensajes por la salida estándar (consola) en modo texto.

Esta versión es plenamente funcional, pero adolece de falta de depuración (hay bastantes fallos), y se puede encontrar, al igual que la versión actual, en
https://sourceforge.net/projects/underthecarpet/

5.2 Última versión: v0.2

El objetivo era que fuese multiplataforma. Al menos debía funcionar sobre Windows (API win32) y Linux (API Posix).
Sus principales características, relacionadas con la entidad que agrupa su funcionalidad son:

- Controller: Es posible cambiar de Presentation.

- Media: Maneja más ficheros de texto (DOC, RTF).
- Modules: Detecta además los programas ImageHide, InThePicture, InvisibleSecrets, JpegX, PGE, SqFileHide, Steganography1.5 y Steganography1.6. Aumentamos Toolkit con get_lsbs; get_2_lsbs; hex_to_int; bool_to_int; bool_to_char; reverse; y copy_path.
- Presentation: Puede mostrar mensajes que no sean de error.
- Parser: Permite la lectura de directorios.
- Lib: Versiones Windows y Linux de jpeglib.

Aunque funciona correctamente, consume demasiados recursos (la memoria ocupada crece linealmente con el número de ficheros analizados), y la tasa de falsos positivos/negativos de algunos módulos es más alta de la esperada.

5.3 Futuras versiones

Más adelante tenemos planeado mejorar y aumentar las capacidades de UTC:

- v0.3: El objetivo es alcanzar una ejecución eficiente y libre de errores (graves). Se añadirá un método de recolección de estadísticas (qué porcentaje de los ficheros analizados dieron positivo, con qué módulos, tiempo empleado en la detección…) y se intentará completar los módulos de detección en imágenes hasta la totalidad de los programas conocidos.
- v0.4: El objetivo será completar los medios sobre los que actuamos. En concreto, se añadirá detección sobre sonido (MP3 y WAV). También se puede integrar la nueva versión de StegDetect (v0.6) y añadir una interfaz gráfica de ventanas.
- v0.5: El objetivo es implementar mecanismos de detección innovadores, como el PoV (pair of values), blind attack, visual attack o cualquier otro de las publicaciones científicas.
- En el futuro, se puede completar con la recuperación del mensaje oculto, e incluso con su desciframiento si fuese necesario. Pero esto ya es otra historia…

SECCIÓN II:
LOS MÓDULOS IMPLEMENTADOS EN UTC

SECCIÓN 2.1
AÑADIR DATOS AL FINAL DEL FICHERO

DOCUMENTACIÓN MÓDULO CAMOUFLAGE

Lista de Contenidos

1. Explicación del módulo

Se trata de una técnica de esteganografía que esconde datos detrás de otros ficheros con extensión JPG.Esta técnica se encuentra dentro del tipo de software esteganográfico que añade los datos al final del fichero.Casi todas las ficheros con un formato determinado tienen una estructura fija de manera que es muy sencillo encontrar cuál es el final de un fichero.

Se trata de una técnica usada en algunos softwares esteganográficos , pero es un método muy débil y fácil de detectar porque se trata tan sólo de buscar un patrón de búsqueda dentrás del final del fichero visible, por tanto es evidente detectarlo y en el caso de Camouflage estos datos no están encriptados y si usan passwords es bastante fácil de detectarla.

A primera vista no la diferencia entre el fichero que tiene datos ocultos y el que no, no es visible a primera vista, este es el objetivo de los métodos de Esteganografía.He aquí dos ficheros uno con datos ocultos y otro sin ellos donde no es posible apreciar la diferencia:

Fichero con datos ocultos 

 **Fichero sin datos ocultos**

2. Motivos para hacerlo

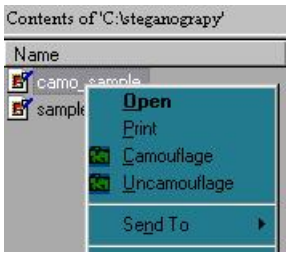
La Esteganografía puede tratarse como algoritmos muy complicados de detectar o como sencillos mecanismos de ocultamiento de información como esconder los datos sin encriptación ni password,como por ejemplo con el Camouflage.Evidentemente si queremos esconder información importante que no queremos sea detectada no usaremos este tipo de software, pero la esteganografía también puede ser usada para el envio de ficheros con información oculta que es inocente,entre amigos o usuarios poco avanzados, y en ese caso este tipo de técnica es bastante útil porque es muy sencilla de usar y detectar.

3. Especificación

Para implementar la técnica de Camouflage sólo necesitamos un fichero de entrada JPG , que como la mayoría de los ficheros con este formato tienen su final del fichero en una marca concreta(FFD9) .Esto no es una regla ,pueden existir ficheros JPG que no terminen en esta cadena pero después de bajarnos varios ficheros JPG hemos observado que casi siempre terminan en este formato.

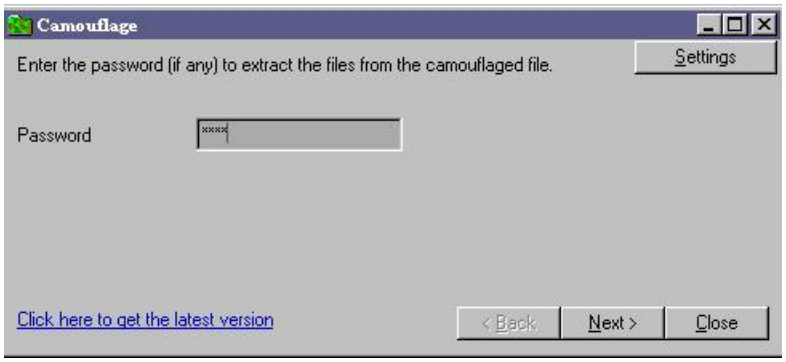
El dato oculto puede ser un simple fichero de texto,otro fichero JPG,un mensaje ASCII.El dato puede tener password o no tenerla y la password en caso de tenerla se añadirá como un código ASCII en un lugar concreto después de la marca que indica que el fichero está usando Camouflage.

Lanzamos el fichero Camouflage y tratamos el fichero JPG en el que hemos insertado la información oculta:



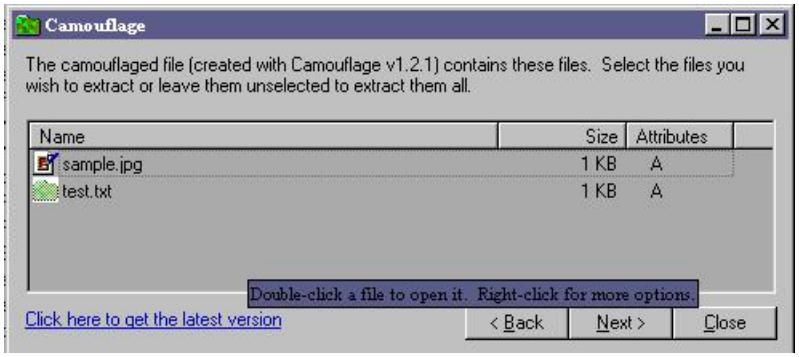
Este es un ejemplo recuperación de datos ocultos a través de Camouflage:

Suponiendo que anteriormente hemos ocultado el archivo mediante el programa Camouflage y le hemos insertado una password.



Si la password no es correcta te saldrá el siguiente mensaje de error:

El resultado serán dos ficheros,el que veíamos anteriormente y el nuevo fichero que se mantenía oculto:



4. Desarrollo del módulo

Si obtenemos un fichero JPG con datos ocultos mediante Camouflage y lo abrimos con editor hexadecimal, veremos que al final de la marca FFD9 tenemos el comienzo del fichero oculto,que empieza con el carácter hexadecimal 2000.

Detrás de la marca los datos pueden estar o no encriptados,la secuencia es primero FFD9 después buscamos 2000 que en decimal “20” sería 32,que es el código ASCII

para el espacio ,seguidos de los datos encriptados o no y al final una firma.This is an example of a file with datos escondidos por Camouflage:

```
1190     D2 46 97 A7 36 4B 11 FE E5 88 F5 5C DF F2 5F FF     Final del fichero JPG
11A0     D9 20 00 10 5E C2 01 B0 6C B1 38 10 5E C2 01 50     Comienzo de datos ocultos
data
11B0     B6 E7 88 10 5E C2 01 00 34 6A 25 1B 00 00 00 56
11C0     FD 13 51 2C CF 67 C1 95 A7 DA 45 53 0A FD C1 FC
11D0     11 6A 3E 9E 85 06 35 CA 46 E3 FF FF FF FF 20 00     Datos
11E0     10 5E C2 01 20 12 83 53 10 5E C2 01 80 3D E9 88     + fichero datos encriptados
11F0     10 5E C2 01 90 22 3F 72 71 F0 19 50 69 D2 4B 8C
1200     84 BC CC 04 47 0A B0 C7 E1 11 20 20 20 20 20 20
1210     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1220     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1230     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1240     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1250     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20     Buffer vacio
1260     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1270     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1280     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1290     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
12A0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
12B0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
12C0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
12D0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
12E0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
12F0     20 20 20 20 20 20 20 67 F8 0A 56 75 88 7E 91 86     Datos
1300     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1310     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1320     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1330     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1340     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1350     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1360     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20     Buffer vacio
1370     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1380     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1390     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
13A0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
13B0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
13C0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
13D0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
13E0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
13F0     20 20 20 20 20 20 1B 00 00 00 A1 11 00 00 02 00     Datos
1400     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1410     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1420     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1430     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1440     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1450     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1460     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20     Buffer vacio
1470     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1480     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1490     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14A0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14B0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14C0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14D0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14E0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14F0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 74
1500     A4 54 10 22 97 20 20 20 20 20 20 20 20 20 20 20     Datos
1510     20 20 20
```

Se podría detectar cuál de dichos campos nos da información extra del fichero como por ejemplo su tamaño , pero no es necesaria más información para detectar la información oculta.

Ahora ponemos un ejemplo de fichero con password:

```
1400     63 F4 1B 43 20 20 20 20 20 20 20 20 20 20 20 20
1410     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1420     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1430     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1440     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1450     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1460     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1470     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1480     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
1490     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14A0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14B0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14C0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14D0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14E0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
14F0     20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
```

Después de observar varios ficheros modificados con Camouflage que tienen password nos damos cuenta que cuando la password cambia el valor encriptado,es decir, que la encriptación no depende de la password.Solo la password consiste en la modificación de los primeros bytes que observamos arriba, por tanto no es muy complicado obtenerla.

5. Forma de uso en UTC

En UTC Camouflage ha sido implementado de manera que tenemos que insertar el fichero que creemos puede estar infectado y pasarle el analizador de Camouflage que nos dirá si se trata de un fichero con información oculta o no.

Damos por hecho que es posible que haya ficheros que detrás del FFD9 tengan basura , pero entendemos que si se encuentra la cadena “2020” que caracteriza al Camouflage se trata del fichero que estamos buscando.

El tratamiento de password no lo hemos incluido por ahora,esperamos implementarlo en futuras versiones.

6. Informes adicionales

Tenemos que tener en cuenta que la búsqueda de información oculta en Camouflage no siempre va a ser fiable, porque esta técnica se basa en al teoría de que la terminación de los ficheros JPEG es siempre FFD9, y esto no es del todo cierto, porque la realidad es que en muchas ocasiones detrás del FFD9 se insertan bits basura que no son relevantes para el resultado final pero que existen.

7. Documentacion adicional

http://camouflage.unfiction.com/

www.jjtc.com/stegoarchive/stego/software.html

www.securitydocs.com/library/1332

DOCUMENTACIÓN MÓDULO JPEGX

Lista de Contenidos

1. Explicación del módulo

Se trata de otro software basado en la misma idea del Camouflage que consiste en añadir información detrás del final del fichero JPG, y al igual que el Camouflage es muy fácilmente detectable y un software que no es recomendado para ocultar información relevante porque es bastante débil a ataques externos.

2. Motivos para hacerlo

Los motivos son los mismos que tuvimos en el Camouflage, aunque este tipo de software es fácilmente detectable y no es muy útil para ocultar información que sea importante mantenerla oculta,se trata de un software que se usa en la red y UTC no sólo está pensado para algoritmos extremadamente complicados sino también para estos tipos de softwares que también se consideran Esteganografía.

3. Especificación

Los ficheros que se usan con Jpegx se tratan de ficheros JPG que suponemos que terminan cuando se detecta la marca FFD9.

Esto es un ejemplo de una imagen con sin estego:

Y la misma imagen con estego usando el método JpegX:

4. Desarrollo del módulo

El nivel de seguridad en los ficheros modificados a través de la técnica JpegX es muy bajo, se basa en las mismas ideas que en el Camouflage pero con ciertas diferencias.

- Los datos son añadidos al final de los ficheros JPG con una marca identificativa que es muy fácil de detectar.
- La encriptación consiste en una sustitución alfabética de los caracteres en un solo byte por lo que el número de posibilidades que tenemos es 256.

Normalmente no es necesaria una password para acceder a los datos ocultos y la encriptación es bastante básica.Al valor del primer byte del caracater ASCII del mensaje secreto le sumamos 187 y se convierte en “BB” en valor hexadecimal.Al segundo byte se le añade 188, al tercero 189 y así consecutivamente.Para recuperar el dato lo único que tendremos que hacer es el proceso inverso,restando al primer byte 187,al segundo 188 y así consecutivamente hasta conseguir el dato completo.

El proceso de encriptación si usa password consiste en sumar todos los valores de la password y sumarle 1 para el primer carácter,dos para el segundo carácter y así consecutivamente hasta conseguir la password encriptada,es un proceso bastante sencillo.

Un ejemplo: escondemos el texto "Tuesday at 12!!" con la password "gloup2=", esto es lo que JpegX va a hacer:

Texto	T	u	e	s	d	a	y	.	a	t	.	1	2	.	!	!
ASCII	54	75	65	73	64	61	79	20	61	74	20	31	32	20	21	21
Suma BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB	BB
Suma Increm	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Suma Carry	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0
Resultado	10	32	23	32	24	22	3B	e2	25	39	E5	F7	F9	E8	EA	EB

Este es el valor final si no usamos la pasword si ahora le añadimos el valor de la ASCCI de la password:

"gloup2=" en ASCII: 67 6C 6F 75 70 32 3D

Si lo añadimos obtenemos el valor 99 en hexacecimal, esta es la clave de acceso y ahora le añadimos a cada byte del texto resultado de la primera tabla un crecimiento de 0,1,2,3...:

Texto	10	32	23	32	24	22	3B	e2	25	39	E5	F7	F9	E8	EA	EB
Suma Key	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99	99
Suma Increm	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
Suma Carry	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1
Resultado	A9	CC	BE	CE	C1	C0	DA	83	C6	DB	89	9C	9F	8F	92	94

Un ejemplo de fichero modificado con Jpegx en hexadecimal sin password:

```
1190     D2 46 97 A7 36 4B 11 FE E5 88 F5 5C DF F2 5F FF     Final del fichero JPG
11A0     D9 5B 3B 31 53 00 57 61 72 6E 69 6E 67 21 20 4D     Marca JpegX
11B0     6F 64 69 66 69 63 61 74 69 6F 6E 20 6F 66 20 74
11C0     68 69 73 20 66 69 6C 65 20 77 69 6C 6C 20 72 65
11D0     73 75 6C 74 20 69 6E 20 69 74 20 6E 6F 20 6C 6F
11E0     6E 67 65 72 20 77 6F 72 6B 69 6E 67 2E 20 4A 50
11F0     65 67 58 20 31 2E 30 2E 36 10 00 10 32 23 32 24     Tamaño fichero oculto
1200     22 3B E2 25 39 E5 F7 F9 E8 EA EB                    Texto oculto
```

Otro ejemplo con password:

```
1190     D2 46 97 A7 36 4B 11 FE E5 88 F5 5C DF F2 5F FF     Final del fichero JPG
11A0     D9 5B 3B 31 53 00 57 61 72 6E 69 6E 67 21 20 4D     Marca JpegX
11B0     6F 64 69 66 69 63 61 74 69 6F 6E 20 6F 66 20 74
11C0     68 69 73 20 66 69 6C 65 20 77 69 6C 6C 20 72 65
11D0     73 75 6C 74 20 69 6E 20 69 74 20 6E 6F 20 6C 6F
11E0     6E 67 65 72 20 77 6F 72 6B 69 6E 67 2E 20 4A 50
```

```
11F0     65 67 58 20 31 2E 30 2E 36 10 00 A9 CC BE CE C1     Tamaño fichero oculto
1200     C0 DA 83 C6 DB 89 9C 9F 8F 92 94                    Texto oculto
```

5. Forma de uso en UTC

La implementación del algoritmo es muy similar a la de Camouflage variando el patrón de búsqueda que caracteriza a los ficheros que usan JpegX,en este caso 5B3B315300.Es posible que se produzcan falsos positivos a la hora de analizar diferentes ficheros de la red, debido a que no hemos implementado la posible basura que se puede añadir al final de la marca FFD9 y que supondría que no encontraría la marca que distingue a este programa.

6. Información adicional

JpegX es completamente gratuito pero su seguridad es bastante baja,aquí hay algunas páginas web que hablan sobre la vulnerabilidad de la password con el JpegX:

www.xatrix.org/article.php?s=3072

cert.uni-stuttgart.de/archive/ bugtraq/2003/04/msg00116.html

xforce.iss.net/xforce/xfdb/11733

A la vulnerabilidad de la password tenemos que añadir que JpegX no incluye encriptación de los datos, lo que definitivamente le hace un software débil y de sólo uso para ocultar información no relevante.

7. Documentos adicionales

www.nerdlogic.org/jpegx/

http://fileforum.betanews.com/detail/1034204107/1

http://www3.ca.com/securityadvisor/vulninfo/Vuln.aspx?ID=8398

http://www.filetransit.com/view.php?id=3663

DOCUMENTACIÓN MÓDULO STEGANOGRAPHY

Lista de Contenidos

1.Explicación del módulo

Este programa surgió dentro de un desarrollo de software llamadao SecureKit y se usa también para ficheros JPG y en la idea de añadir datos detrás del final del fichero JPG, suponiendo que el fichero JPG termina siempre en la marca "FFD9", cosa que no se cumple en todos los casos, ya que en muchas ocasiones detrás de ficheros JPG se insertan otros datos y no significa que sean siempre contendios ocultos.

2. Motivos para hacerlo

Los motivos son los mismos que en los anteriores módulos, añadir el máximo número de programas de detección de estegos y así hacer a nuestra aplicación más potente a la hora de buscar contenidos ocultos en Internet sin tener en cuenta si se trata de métodos seguros o no.

3. Especificación

Se usa para ficheros JPG y consiste en tener dos ficheros con los datos de los bits uno detrás de otro mientras sólo es visible uno de ellos y para obtener los datos del segundo es necesario usar el método Steganography para conseguir los datos del segundo.La marca que distingue el primero del segundo es FFD9, igual que en Camouflage y JpegX.

4. Desarrollo del módulo

STEGANOGRAPHY 1.50

Según sus desarrolladores este programa usa una encriptación basada en 256 bits pero hemos comprobado que en la mayoría de los casos esta encriptación no se lleva a cabo y si eres capaz de detectar la presencia de información oculta no es necesario ningún algoritmo de encriptación para obtener los datos , sino tan sólo una password que estará o no definida.Aunque nosotros en la aplicación no hemos llevado a cabo el módulo de recuperación de estos datos, sólo la detección de información oculta.

La información oculta va detrás de la información del fichero visible y comprimida en un fichero ZIP.Este es un ejemplo en hexadecimal de un fichero JPG que usa Steganography sin password:

```
00000090   00 00 00 FF DA 00 08 01  01 00 00 3F 00 AA 60 3F
000000A0   FF D9 50 4B 03 04 14 00  02 00 08 00 9C 41 23 30
000000B0   05 0B 24 E1 0A 00 00 00  0B 00 00 00 11 00 11 00
000000C0   68 69 64 64 65 6E 6D 65  73 73 61 67 65 2E 74 78
000000D0   74 55 54 0D 00 07 D8 BF  F6 3F A0 92 F6 3F D6 BF
000000E0   F6 3F 2B C8 CF C9 CF 55  28 00 91 00 50 4B 01 02
000000F0   17 0B 14 00 02 00 08 00  9C 41 23 30 05 0B 24 E1
00000100   0A 00 00 00 0B 00 00 00  11 00 09 00 00 00 00 00
00000110   00 00 20 00 80 81 00 00  00 00 68 69 64 64 65 6E
00000120   6D 65 73 73 61 67 65 2E  74 78 74 55 54 05 00 07
00000130   D8 BF F6 3F 50 4B 05 06  00 00 00 00 01 00 01 00
00000140   48 00 00 00 4A 00 00 00  00 00 48 49 5A 00 A8 00
00000150   00 00 64 34 31 64 38 63  64 39 38 66 30 30 62 32
00000160   30 34 00 00 03 00
```

Todo lo que hay detrás de FFD9 se considera el mensaje oculto y la parte subrayada sería la password en caso de que tuviera, como en este caso no tenemos Steganography mete bytes de relleno.La marca que distingue a Steganography en esta versión es “PK”.

El mismo fichero con la misma información oculta en hexadecimal pero esta vez con password tendría la siguiente apariencia:

```
00000090   00 00 00 FF DA 00 08 01  01 00 00 3F 00 AA 60 3F
```

```
000000A0   FF D9 50 4B 03 04 14 00  02 00 08 00 9C 41 23 30
000000B0   05 0B 24 E1 0A 00 00 00  0B 00 00 00 11 00 11 00
000000C0   68 69 64 64 65 6E 6D 65  73 73 61 67 65 2E 74 78
000000D0   74 55 54 0D 00 07 D8 BF  F6 3F A0 92 F6 3F D6 BF
000000E0   F6 3F 2B C8 CF C9 CF 55  28 00 91 00 50 4B 01 02
000000F0   17 0B 14 00 02 00 08 00  9C 41 23 30 05 0B 24 E1
00000100   0A 00 00 00 0B 00 00 00  11 00 09 00 00 00 00 00
00000110   00 00 20 00 80 81 00 00  00 00 68 69 64 64 65 6E
00000120   6D 65 73 73 61 67 65 2E  74 78 74 55 54 05 00 07
00000130   D8 BF F6 3F 50 4B 05 06  00 00 00 00 01 00 01 00
00000140   48 00 00 00 4A 00 00 00  00 00 48 49 5A 00 A8 00
00000150   00 00 30 63 63 31 37 35  62 39 63 30 66 31 62 36
00000160   61 38 00 00 03 00
```

Como se puede observar la información oculta tiene los mismos datos, por tanto no existe ningún tipo de encriptación y el campo de la password es lo que está subrayado.

Y este es un ejemplo de un fichero con información oculta con Steganography pero esta vez con password “b”:

```
00000090   00 00 00 FF DA 00 08 01  01 00 00 3F 00 AA 60 3F
000000A0   FF D9 50 4B 03 04 14 00  02 00 08 00 9C 41 23 30
000000B0   05 0B 24 E1 0A 00 00 00  0B 00 00 00 11 00 11 00
000000C0   68 69 64 64 65 6E 6D 65  73 73 61 67 65 2E 74 78
000000D0   74 55 54 0D 00 07 D8 BF  F6 3F A0 92 F6 3F D6 BF
000000E0   F6 3F 2B C8 CF C9 CF 55  28 00 91 00 50 4B 01 02
000000F0   17 0B 14 00 02 00 08 00  9C 41 23 30 05 0B 24 E1
00000100   0A 00 00 00 0B 00 00 00  11 00 09 00 00 00 00 00
00000110   00 00 20 00 80 81 00 00  00 00 68 69 64 64 65 6E
00000120   6D 65 73 73 61 67 65 2E  74 78 74 55 54 05 00 07
00000130   D8 BF F6 3F 50 4B 05 06  00 00 00 00 01 00 01 00
00000140   48 00 00 00 4A 00 00 00  00 00 48 49 5A 00 A8 00
00000150   00 00 39 32 65 62 35 66  66 65 65 36 61 65 32 66
00000160   65 63 00 00 03 00
```

Como puede verse también el único cambio es la zona donde se escribe la password.

A primera vista parece que los datos están en valores hexadecimales pero es fácil apreciar que la conversión de estos valores en ASCII parecen también valores hexadecimales.En definitiva que parece que los valores son almacenados en un formato extraño que deja el caráceter espacio dos veces de manera que parezca que son valores hexadecimales.Por ejemplo el valor “FF” en hexadecimal es guardado como “102 102” o “66 66” que es la representación de “FF” en hexadecimal.

El algoritmo de la password usado es MD5 usando una utilidad gratuita llamada HashCalc.Aquí hay algunos ejemplos:

```
128 bits (16 bytes) MD5 búsqueda de cadena "" es "d41d8cd98f00b204e9800998ecf8427e"

128 bits (16 bytes) MD5 búsqueda de cadena "" es "0cc175b9c0f1b6a831c399e269772661"

128 bits (16 bytes) MD5 búsqueda de cadena "" es "92eb5ffee6ae2fec3ad71c777531578f"
```

Para extraer los datos ocultos “encriptados” o recuperarlos existen dos posibilidades:

-Calcular la búsqueda MD5 de la password y sobrescribir los bytes de la password actual.

-Extraer los datos ocultos en hexadecimal y abrirlos con el Windows de manera que podremos leer sin problemas, ya que como hemos visto anteriormente carece de encriptación.

STEGANOGRAPHY 1.60

Existen dos versiones de este producto 1.50 y 1.60 , la idea para ambos es la misma y la versión superior se diferencia en que el ocultamiento de los datos ocultos como un fichero zip lo oculta de manera que ya no es posible distinguirlo y no poder copiar los valores hexadecimales una vez extraídos como hacíamos con la versión 1.50 pero no añade ninguna seguridad , tan sólo lo hace menos evidente, y el patrón a buscar después del fin de fichero es “HIZ”.

Con Steganography 1.60 el fichero en hexadecimal ahora tendría el siguiente aspecto suponiendo una password “a”:

```
00000290   08 09 42 00 4A 52 30 00  D8 01 A0 3F FF D9 9E 97
000002A0   BA 2A 00 80 88 C9 A3 70  97 5B A2 E4 99 B8 C1 78
000002B0   72 0F 88 DD DC 34 2B 4E  7D 31 7F B5 E8 70 39 A8
000002C0   B8 42 75 68 71 91 03 5F  A4 A4 E0 EA 43 9F AA 8E
000002D0   04 99 FA A5 EC 63 9F 23  79 7E 5D 91 33 43 3B C3
000002E0   C1 C4 21 5F 71 5F DF C5  D9 1C C5 DE 83 6C 4D EC
000002F0   D5 DC 45 A9 D7 C7 09 EE  15 21 53 28 0E 56 4A A9
00000300   22 71 C1 03 3E D5 F0 EE  68 F5 4F DC 08 01 F3 61
00000310   34 41 E7 10 89 75 89 30  C0 49 7A CF C6 CB 11 B4
00000320   F1 10 19 93 83 27 E2 8B  49 26 33 F3 76 8E A4 11
00000330   D2 7D A6 17 96 F9 11 58  02 E2 4C A3 B5 32 33 32
00000340   21 B1 1A A6 D9 FD 23 45  1F 87 13 3F 45 34 E0 8C
00000350   F1 E5 0D 35 A9 5C EB DD  F3 EC 32 5F DE 38 93 86
00000360   8A D7 F8 5D EB B9 FC 1C  E0 68 D7 C2 91 68 B1 DC
00000370   44 6C E2 DC 27 E1 A3 5B  71 29 8B 78 E5 04 D5 6E
00000380   81 CF 34 BF 1F 5B E8 55  B1 C2 7F F3 62 F0 48 49   HIZ
00000390   5A 00 F0 00 00 00 38 6B  6B 39 3F 3D 6A 31 6B 38
000003A0   6E 39 6A 3E 69 30 00 00  03 00
```

Ahora podemos ver que el fichero formateado en zip está ahora encriptado , de forma que la estructura no es visible pero al final de los datos podemos ver el valor HIZ en hexadecimal y detrás otros 16 bytes como en los casos anteriores.

En la versión 1.50 estos bytes eran la mitad de la la password en MD5 :

Steganography 1.50: 30 63 63 31 37 35 62 39 63 30 66 31 62 36 61 38 0cc175b9c0f1b6a

Steganography 1.60: 38 6B 6B 39 3F 3D 6A 31 6B 38 6E 39 6A 3E 69 30 8kk9?=j1k8n9j>i0

Sin password teníamos lo siguiente:

Steganography 1.50: 64 34 31 64 38 63 64 39 38 66 30 30 62 32 30 34 d41d8cd98f00b204

Steganography 1.60: 6C 3C 39 6C 30 6B 6C 31 30 6E 38 38 6A 3A 38 3C l<9l0kl10n88j:8<

Observando las dos versiones parece que el método de encriptación tan sólo consiste en añadir 8 a cada byte y este es el campo que el programa usa para verificar que la password que el usuario introduce es correcta.Por tanto los datos ocultos no están ncriptados en función de la password y no la necesitaríamos a la hora de obtener los datos encriptados.Esto significa que el primer método que usamos en la versión 1.50 es todavía útil y basta con calcular la password con MD5 para obtener tan sólo la mitad y añador 8 a cada byte de la representación ASCII de los valores hexadecimales y sobreescribirlos en la estructura HIZ al final del fichero camuflado.

5. Forma de uso en UTC

En nuestra aplicación no hemos desarrollado el módulo de implementación de recuperación de datos ocultos, al igual que en los anteriores módulos hemos visto sólo la detección de los datos y devolvemos un valor positivo o negativo según haya o no información oculta.

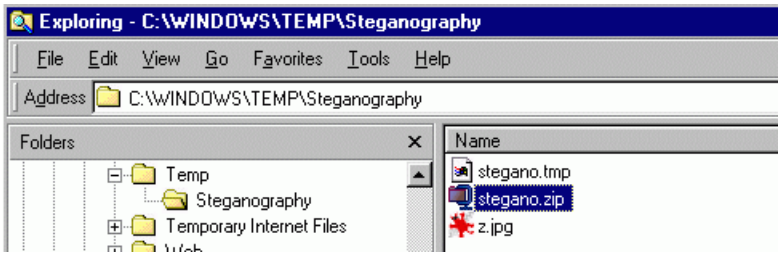
6. Informes adicionales

Este programa tiene varios problemas de seguridad.Muchos programas usan fichero temporales para trabajar que se encargan de calcular algún dato y evitan usar la memoria y se crea un fichero en el disco duro y escriben en ellos que después son borrados.Los programas de encriptación normalmente intentan evitar este tipo de ficheros , especialmente la escritura en ficheros temporales de passwords o datos encriptados porque eso supone un agujero de seguridad bastante grande a la hora de posibles atacantes.

Porque los ficheros borrados no están autenticamente borrados , sino que desaparecen de la lista de directorios pero su contenido está todavía en el disco duro y dependiendo de la cantidad de espacio de la que dispongas pueden permanecer en el disco más o menos tiempo.

Los buenos programas de encriptación lo que hacen es escribir los ficheros temporales que se van creando de manera que sobreescriben los viejos y de esa manera no se borran y no permanecen en el disco duro y los viejos ficheros temporales desparecen por completo de nuestro soporte magnético.

Steganography en este sentido no se considera un buen método de encriptación porque no tiene en cuenta este aspecto y puedes chequear en el fichero temporal de windows que los ficheros existen y ver cuál es su nombre para más tarde buscarlos en el disco duro:



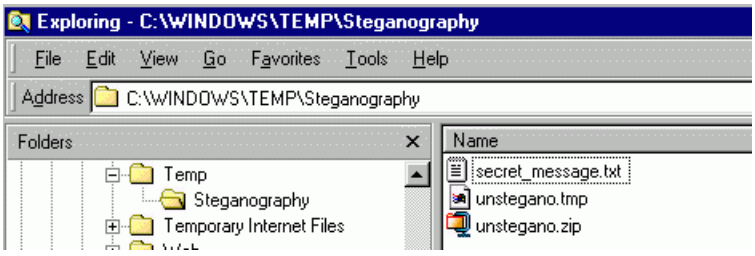
Aquí vemos los tres ficheros que al principio genera Steganography en el directorio temporal:

1.El fichero temporal

2.El fichero con la información oculta en formato ZIP

3.La imagen visible que esconde el fichero oculto.

Cuando desencriptamos el fichero nos queda lo siguiente:



En los dos casos el fichero con la información oculta está duplicado en un formato zip o en un formato plano.Cuando el programa borra estos ficheros su contenido está todavía ahí pudiendo recuperarlos con algún programa que te permite recuperar datos borrados del disco duro.

7. Documentos adicionales

www.fileheaven.com/Steganography/download/9270.htm

www.dailytools.com/download-1162.html

www.downloadplanet.net/SubCat62-500.htm

www.downloadfreetrial.com/utilities/util14000.html

DOCUMENTACIÓN MÓDULO SQFILEHIDE

Lista de Contenidos

1.Explicación del módulo

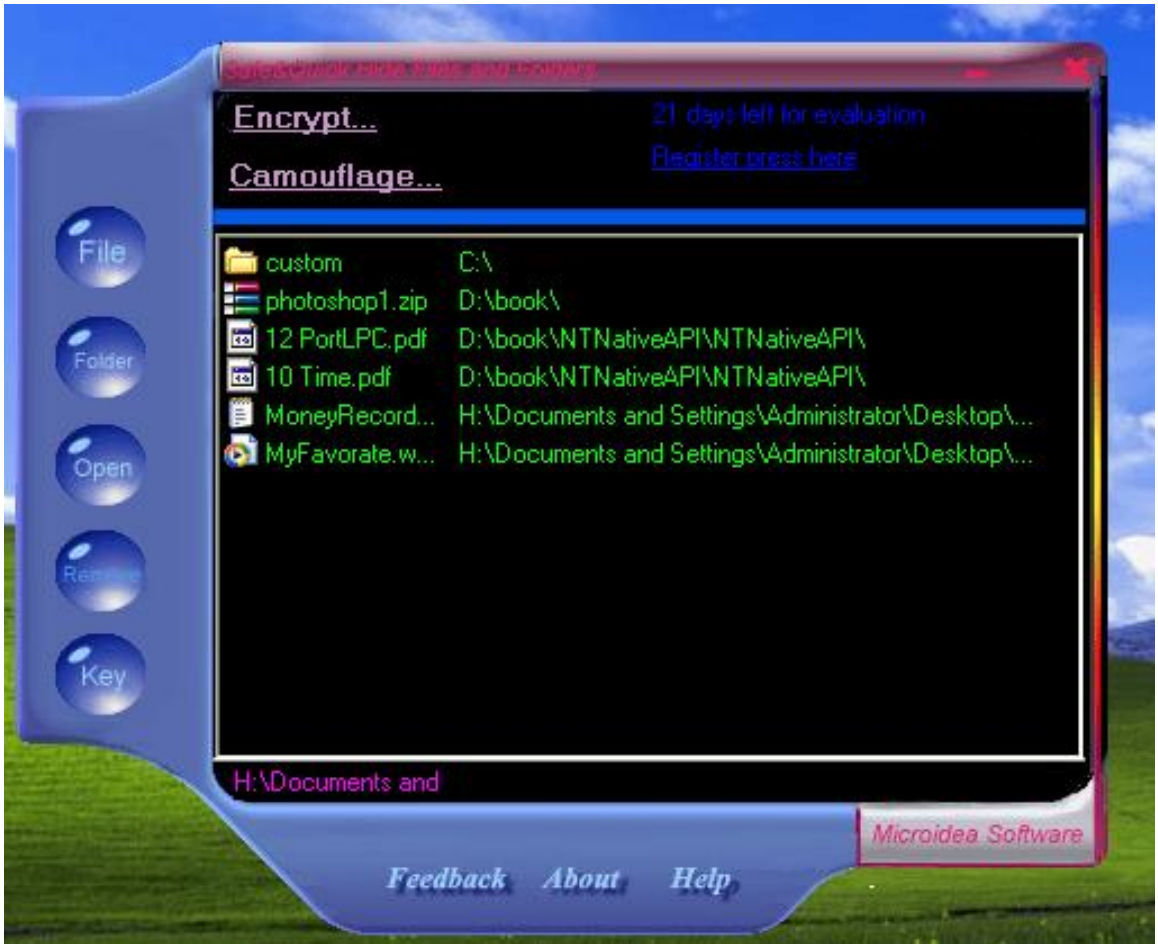
El software fue vendido por la empresa MicroIdea y consiste en esconder ficheros desde el explorador de Windows haciéndolos más difíciles de encontrar.Es bastante fácil de usar con una interfaz sencilla y clara.Como consiste también en el tipo de módulos que añaden información al final del fichero JPG , no se trata de un software esteganográfico muy robusto contra posibles ataques , pero éste tiene algunas particularidades que veremos más tarde.

2.Motivos para hacerlo

Es también un software bastante débil como los que hemos visto hasta ahora pero es muy útil ver las técnicas que usa y quizá nos sirva para entender mejor futuros softwares esteganograficos de más difícil detección.

3.Especificación

Se usa para ficheros JPG y su interfaz es muy fácil de usar , la ejecución del programa tiene el siguiente aspecto:



4.Desarrollo de módulo

Como este software se encuentra dentro de las técnicas de esteganografía que añaden información al final del fichero, podemos sacar los valores del fichero JPG a valores hexadecimales con un editor y analizar su contenido.Este es un ejemplo:

```
00000000   FF D8 FF E0 00 10 4A 46  49 46 00 01 01 01 00 48
00000010   00 48 00 00 FF DB 00 43  00 06 04 05 06 05 04 06
00000020   06 05 06 07 07 06 08 0A  10 0A 0A 09 09 0A 14 0E
00000030   0F 0C 10 17 14 18 18 17  14 16 16 1A 1D 25 1F 1A
00000040   1B 23 1C 16 16 20 2C 20  23 26 27 29 2A 29 19 1F
00000050   2D 30 2D 28 30 25 28 29  28 FF DB 00 43 01 07 07
00000060   07 0A 08 0A 13 0A 0A 13  28 1A 16 1A 28 28 28 28
00000070   28 28 28 28 28 28 28 28  28 28 28 28 28 28 28 28
00000080   28 28 28 28 28 28 28 28  28 28 28 28 28 28 28 28
00000090   28 28 28 28 28 28 28 28  28 28 28 28 28 28 FF C0
```

```
000000A0   00 11 08 00 70 00 5A 03  01 22 00 02 11 01 03 11
000000B0   01 FF C4 00 15 00 01 01  00 00 00 00 00 00 00 00
000000C0   00 00 00 00 00 00 00 08  FF C4 00 14 10 01 00 00
000000D0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 FF C4
000000E0   00 14 01 01 00 00 00 00  00 00 00 00 00 00 00 00
000000F0   00 00 00 00 FF C4 00 14  11 01 00 00 00 00 00 00
00000100   00 00 00 00 00 00 00 00  00 00 FF DA 00 0C 03 01
00000110   00 02 11 03 11 00 3F 00  95 00 00 00 00 00 00 00
00000120   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000130   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000140   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000150   00 00 00 00 00 00 00 00  7F FF D9 50 4B 03 04 14
00000160   00 02 00 08 00 9C B3 84  2F EB 42 27 A2 10 00 00
00000170   00 0E 00 00 00 12 00 00  00 68 69 64 64 65 6E 5F   Mensaje oculto
00000180   6D 65 73 73 61 67 65 2E  74 78 74 0B FA 96 9D 97
00000190   99 97 AE 90 F2 2D 25 B1  48 11 00 50 4B 01 02 00
000001A0   00 14 00 02 00 08 00 9C  B3 84 2F EB 42 27 A2 10
000001B0   00 00 00 0E 00 00 00 12  00 00 00 00 00 00 00 00
000001C0   00 00 00 00 00 00 00 00  00 68 69 64 64 65 6E 5F
000001D0   6D 65 73 73 61 67 65 2E  74 78 74 50 4B 05 06 00
000001E0   00 00 00 01 00 01 00 40  00 00 00 40 00 00 00 00
000001F0   00 5B 01 00 00 96 00 00  00 CC 99 FF 66
```

A partir del FFD9 tenemos el mensaje oculto marcado en rojo.Este mensaje está en formato ZIP igual que con el método Steganography y habrá que buscar después del fin de fichero el patrón “PK”.Se pueden recuperar fácilmente copiando los bytes a un fichero y abriéndolo con el un programa descompresor como el Winzip , porque los datos no están encriptados.

5. Forma de uso en UTC

La implementación del algoritmo es muy similar a la de Camouflage y JpegX variando el patrón de búsqueda que caracteriza a los ficheros que usan SqFileHide,en este caso 504B.Es posible que se produzcan falsos positivos a la hora de analizar diferentes ficheros de la red, debido a que no hemos implementado la posible basura que se puede añadir al final de la marca FFD9 y que supondría que no encontraría la marca que distingue a este programa.

6.Informes adicionales

Si olvidas la password tienen la peculiaridad de dando a una de las opciones de la tabla de Diálogos te envían tu password a tu cuenta de correo,esto significa que en algún lugar se encuentra la password en un fichero plano, lo que significa un agujero negro en la seguridad porque este fichero plano con tu password que se manda a través de la red es visible para todo el mundo.

Una vez que un atacante consigue tu password , todos los algoritmos de encriptación más fuertes o más difíciles de detectar carecen de importancia porque el atacante puede

acceder a los datos sin ningún problema introduciendo la password que ha conseguido de una manera sencilla.Tu puedes ver cuál es el fichero que SqFileHide abrió cuando estaba chequeando la validez de tu password ,después ese fichero se borrará pero sólo desaparece de tu explorador porque en algún lugar del disco duro sigue existiendo y puedes recuperarlo con relativa facilidad.

7.Documentos adicionales

www.tucows.com/preview/295425.html

www.downlinx.com/proghtml/412/41283.htm

www.hotdownloads.com/index.php3?job=3&id=48915

www.microidea.net/SQHideFile/Introduce.htm

DOCUMENTACIÓN MÓDULO PGE

Lista de Contenidos

1.Explicación del módulo

Se trata de una técnica de esteganografía que esconde datos detrás de otros ficheros con extensión JPG.Esta técnica se encuentra dentro del tipo de software esteganográfico que añade los datos al final del fichero.Casi todas las ficheros con un formato determinado tienen una estructura fija de manera que es muy sencillo encontrar cuál es el final de un fichero.

Se trata de una técnica usada en algunos softwares esteganográficos , pero es un método muy débil y fácil de detectar porque se trata tan sólo de buscar un patrón de búsqueda dentrás del final del fichero visible, por tanto es evidente detectarlo y en el caso de Camouflage estos datos no están encriptados y si usan passwords es bastante fácil de detectarla.

2.Motivos para hacerlo

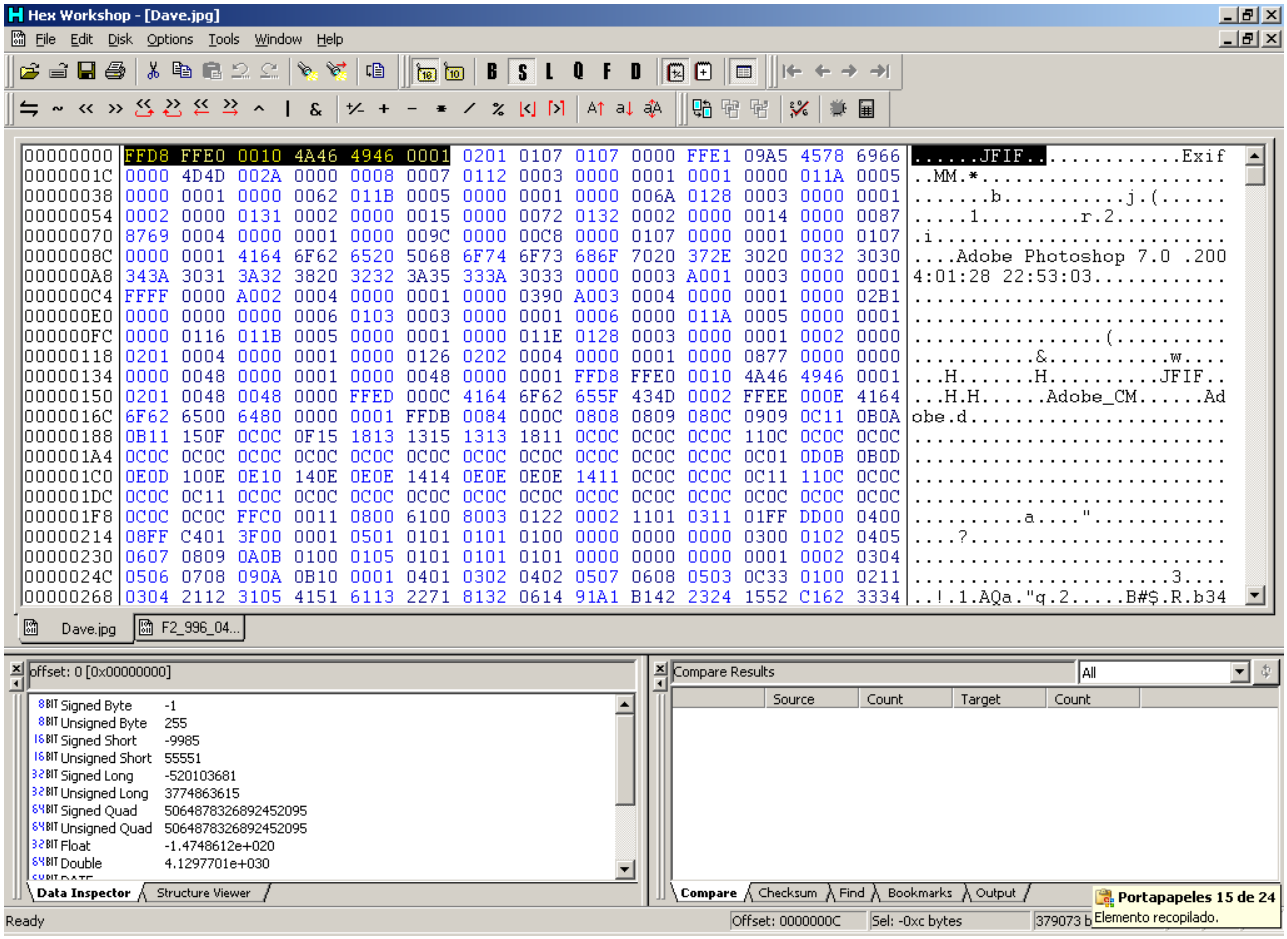
Añadir más funcionalidades a nuestra aplicación.

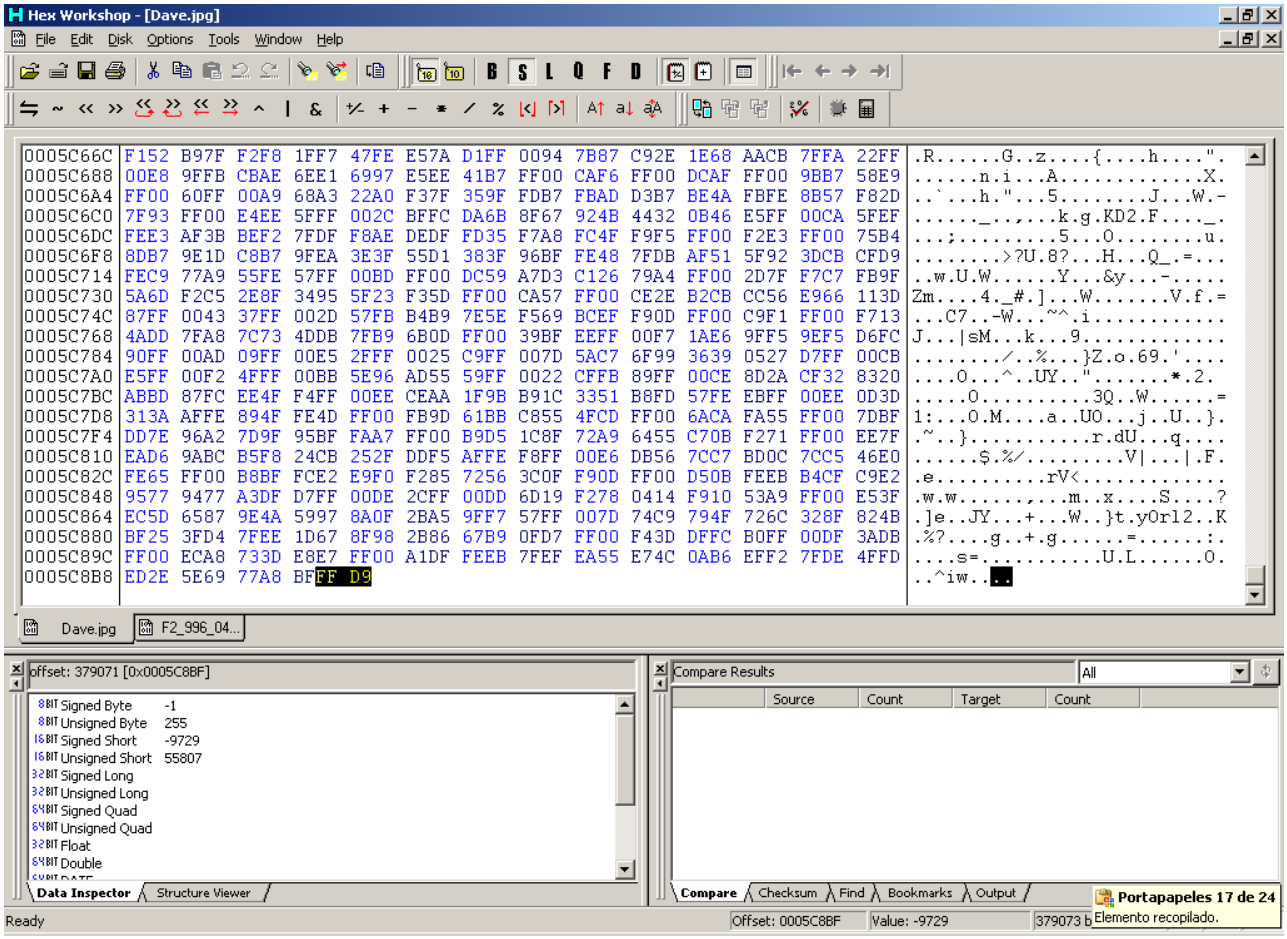
3.Especificación

A continuación, procederemos a comentar cuales han sido los resultados obtenidos en el análisis realizado en la herramienta PGE versión 1.0, la cual ha sido rota en su totalidad.

4.Desarrollo del módulo

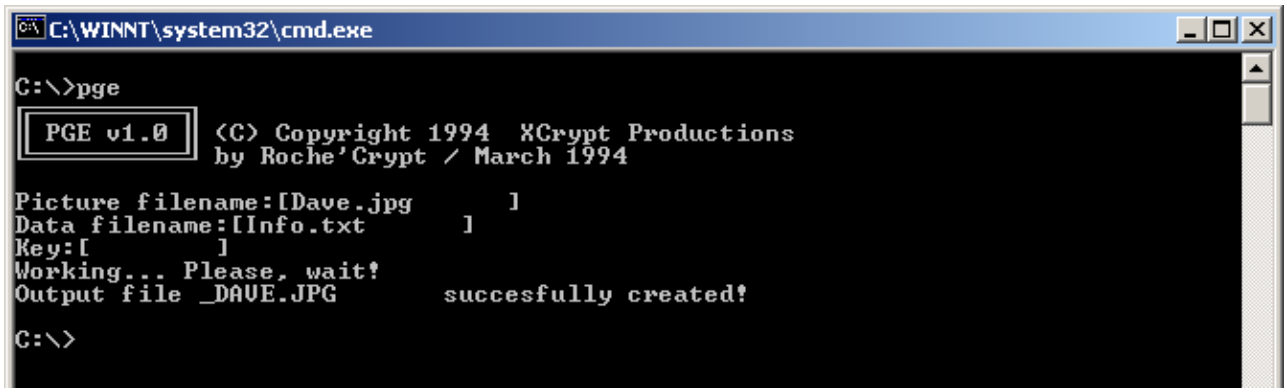
Primeramente analizamos con un editor hexadecimal el contenido de la fotografía JPG que contendría posteriormente la información:


Hex Workshop - [Dave.jpg]
File Edit Disk Options Tools Window Help
00000000 FFD8 FFE0 0010 4A46 4946 0001 0201 0107 0107 0000 FFE1 09A5 4578 6966 ......JFIF..........Exif
0000001C 0000 4D4D 002A 0000 0008 0007 0112 0003 0000 0001 0001 0000 011A 0005 ..MM.*..........
00000038 0000 0001 0000 0062 011B 0005 0000 0001 0000 006A 0128 0003 0000 0001 ....b.....j.(.....
00000054 0002 0000 0131 0002 0000 0015 0000 0072 0132 0002 0000 0014 0000 0087 .....1.....r.2.....
00000070 8769 0004 0000 0001 0000 009C 0000 00C8 0000 0107 0000 0001 0000 0107 .i..............
0000008C 0000 0001 4164 6F62 6520 5068 6F74 6F73 686F 7020 372E 3020 0032 3030 ....Adobe Photoshop 7.0 .200
000000A8 343A 3031 3A32 3820 3232 3A35 333A 3033 0000 0003 A001 0003 0000 0001 4:01:28 22:53:03.....
000000C4 FFFF 0000 A002 0004 0000 0001 0000 0390 A003 0004 0000 0001 0000 02B1 ....
000000E0 0000 0000 0000 0006 0103 0003 0000 0001 0006 0000 011A 0005 0000 0001 ....
000000FC 0000 0116 011B 0005 0000 0001 0000 011E 0128 0003 0000 0001 0002 0000 .....(.....
00000118 0201 0004 0000 0001 0000 0126 0202 0004 0000 0001 0000 0877 0000 0000 .....&.....w....
00000134 0000 0048 0000 0001 0000 0048 0000 0001 FFD8 FFE0 0010 4A46 4946 0001 ...H.....H.....JFIF..
00000150 0201 0048 0048 0000 FFED 000C 4164 6F62 655F 434D 0002 FFEE 000E 4164 ...H.H.....Adobe_CM.....Ad
0000016C 6F62 6500 6480 0000 0001 FFDB 0084 000C 0808 0809 080C 0909 0C11 0B0A obe.d.....
00000188 0B11 150F 0C0C 0F15 1813 1315 1313 1811 0C0C 0C0C 0C0C 110C 0C0C 0C0C .....
000001A4 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C01 0D0B 0B0D .....
000001C0 0E0D 100E 0E10 140E 0E0E 1414 0E0E 0E0E 1411 0C0C 0C0C 0C11 110C 0C0C .....
000001DC 0C0C 0C11 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C 0C0C .....
000001F8 0C0C 0C0C FFC0 0011 0800 6100 8003 0122 0002 1101 0311 01FF DD00 0400 .....a....".....
00000214 08FF C401 3F00 0001 0501 0101 0101 0100 0000 0000 0000 0300 0102 0405 ....?.....
00000230 0607 0809 0A0B 0100 0105 0101 0101 0101 0000 0000 0000 0001 0002 0304 .....
0000024C 0506 0708 090A 0B10 0001 0401 0302 0402 0507 0608 0503 0C33 0100 0211 .....3....
00000268 0304 2112 3105 4151 6113 2271 8132 0614 91A1 B142 2324 1552 C162 3334 ..!.1.AQa."q.2.....B#$.R.b34
Dave.jpg
F2_996_04...
offset: 0 [0x00000000]
8 BIT Signed Byte -1
8 BIT Unsigned Byte 255
16 BIT Signed Short -9985
16 BIT Unsigned Short 55551
32 BIT Signed Long -520103681
32 BIT Unsigned Long 3774863615
64 BIT Signed Quad 5064878326892452095
64 BIT Unsigned Quad 5064878326892452095
32 BIT Float -1.4748612e+020
64 BIT Double 4.1297701e+030
Data Inspector
Structure Viewer
Compare Results
All
Source
Count
Target
Count
Compare
Checksum
Find
Bookmarks
Output
Portapapeles 15 de 24
Elemento recopilado.
Ready
Offset: 0000000C
Sel: -0xc bytes
379073 b

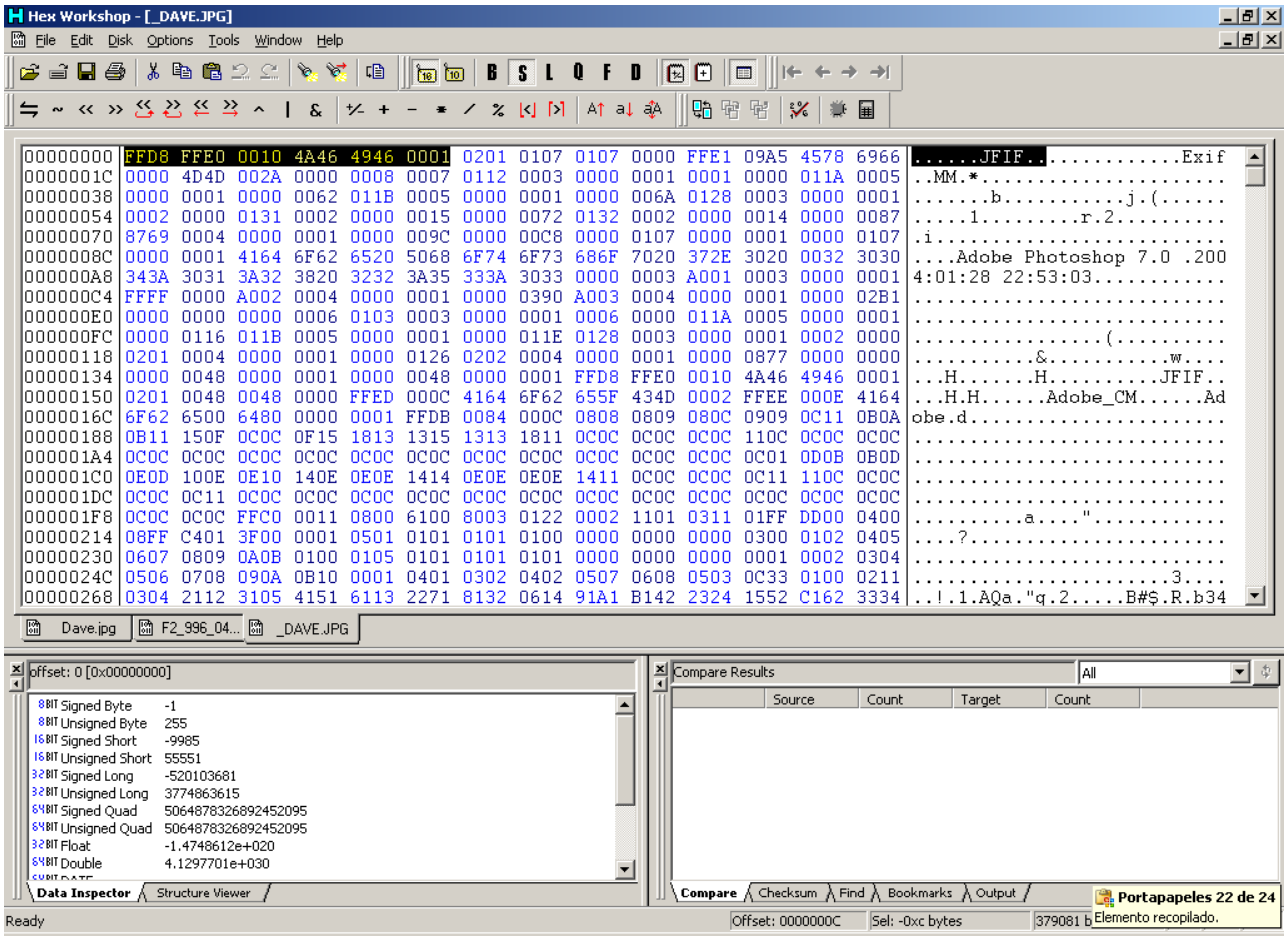


Como se puede observar en los dos pantallazos insertados, el principio del JPG es el que se suele utilizar como estándar “FFD8FFE000104A4649460001” y lo mismo ocurre con el final del fichero JPG, “FFD9”.

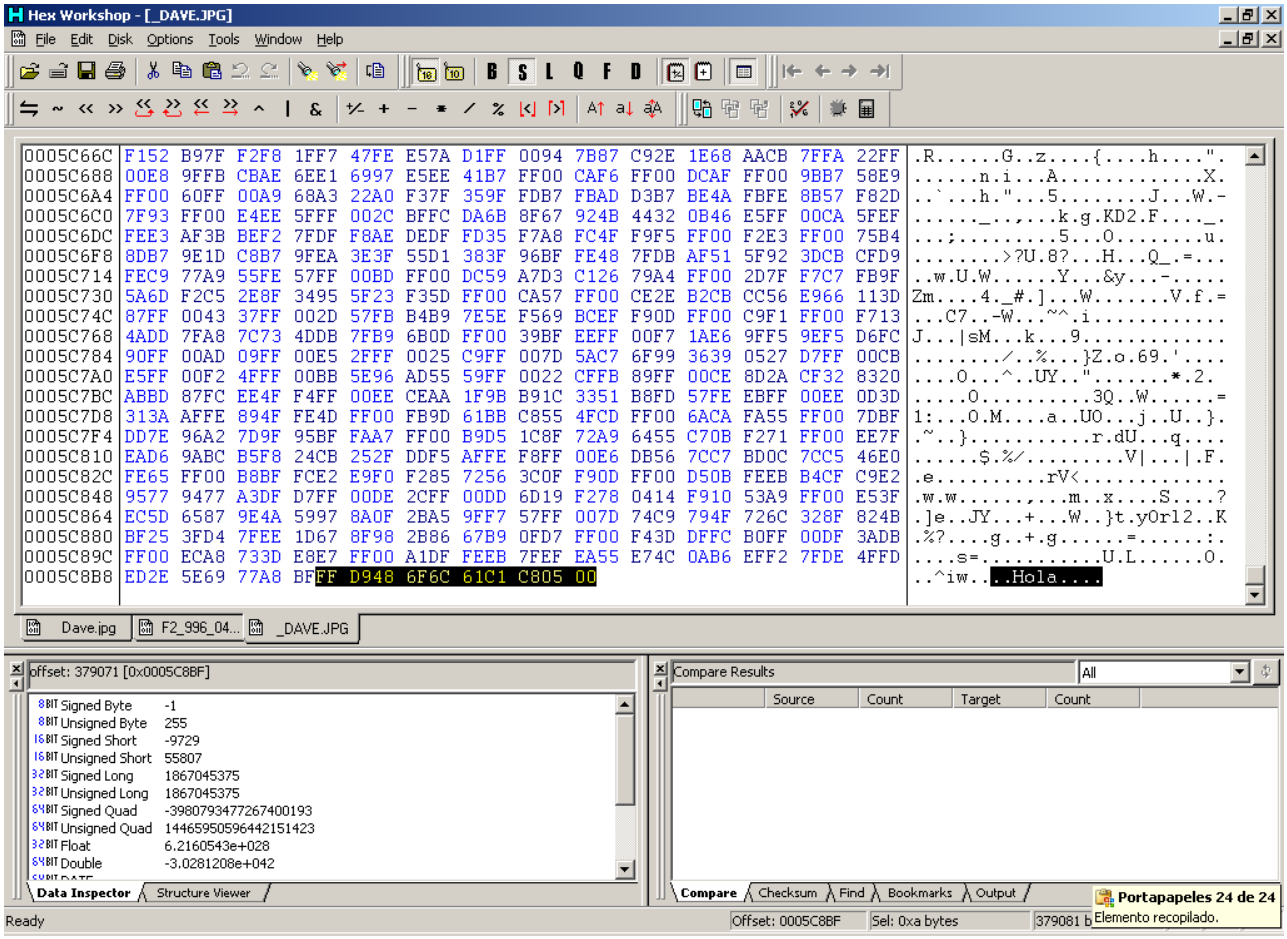
Una vez analizada la fotografía, procedemos a insertar el contenido del fichero “Info.txt” (recordemos que tan solo posee el texto “Hola”) en la fotografía utilizada para la prueba (Dave.jpg) a través de la herramienta PGE v1.0:



Una vez procesado los ficheros Dave.jpg e Info.txt por PGE 1.0, obtenemos como resultado un fichero llamado _Dave.jpg que, en aspecto, si observamos la fotografía directamente, al ojo humano le es idéntico que la fotografía utilizada para ocultar la información. Si hacemos un pequeño análisis hexadecimal obtenemos los siguientes resultados:



Como podemos observar en esta captura de pantalla, el comienzo del fichero JPG se mantiene inalterado, de hecho, el fichero no tiene ninguna modificación hasta el final del fichero. PGE 1.0 no añade ningún tipo de información ni al principio ni entre medias del fichero utilizado para ocultar la información, si no que opera insertando información al final del fichero.



Como se puede observar en la fotografía adjunta, vemos que a continuación de la cadena que indica el final de fichero de un JPG (“FFD9”) se ha insertado información. Después de realizar diversas pruebas hemos llegado a la conclusión de que el sistema siempre hace lo siguiente:

- Mantiene todo el JPG intacto, desde “FFD8FFE000104A4649460001” hasta “FFD9”.
- Añade al final del JPG (a continuación de FFD9) una cadena con la siguiente estructura:
    - Texto a ocultar + “C1C80500”.

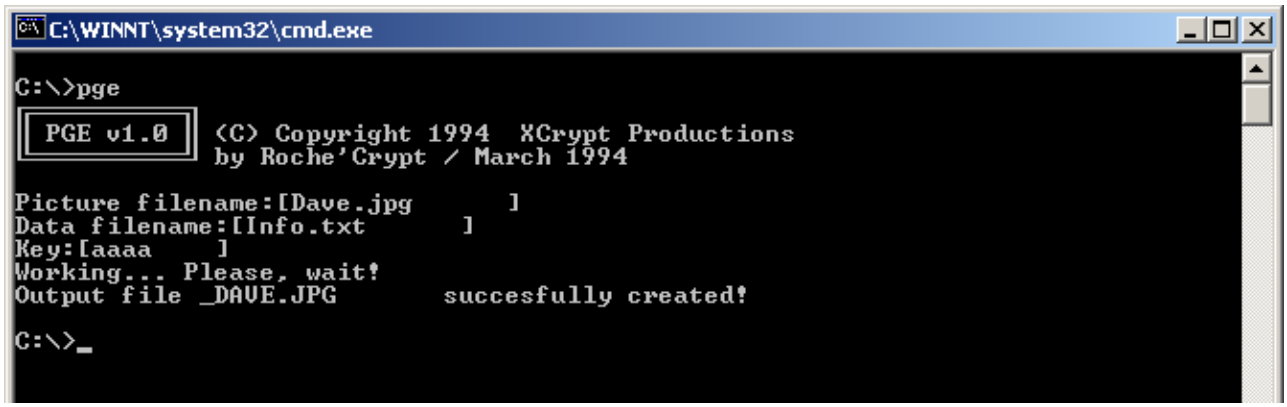
Esta última cadena, “C1C80500”, le sirve a PGE para conocer cual es el final de su propio fichero en el que ha ocultado la información, por lo que se sabe que siempre la información se oculta entre “FFD9” (final de los JPG´s) y “C1C80500” (final de una imagen tratada por PGE 1.0). Por tanto, en el caso mostrado en la imagen, la información ocultada en la fotografía sería “486F6C61”, que como se ha comentado anteriormente es el cifrado en hexadecimal del fichero Info.txt escondido en la fotografía Dave.jpg.

- **Ruptura del sistema utilizando el cifrado**:

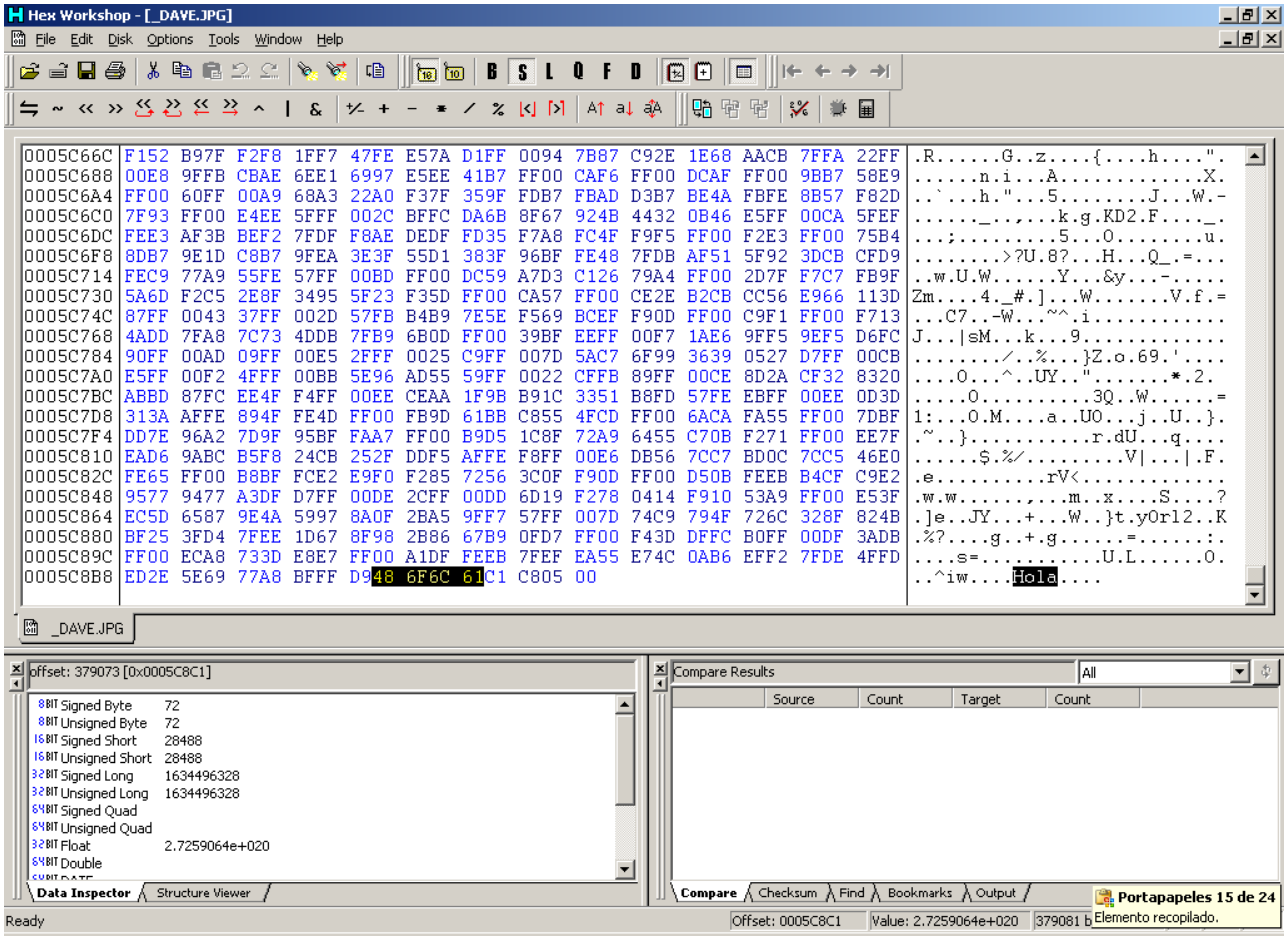
Es importante comentar ciertos aspectos previos del cifrado antes de comentar el proceso seguido para la ruptura del sistema utilizando el cifrado que ofrece PGE v1.0.

El tamaño de las contraseñas a utilizar es como máximo de 8 caracteres y no hace caso a las mayúsculas, es decir, es indiferente cifrar con la palabra "casa" que con la palabra "CASA".

Después de sucesivas pruebas, se detectó que la operación que posiblemente podía utilizar el sistema, era un XOR entre el texto a ocultar en la imagen y la contraseña proporcionada por el usuario. Este razonamiento surgió debido a que, al usar contraseñas con iguales caracteres en un número par, se obtenían operaciones nulas, es decir, no se cifraba el texto que se ocultaba en la imagen como se muestra en la siguiente pantalla:



Como vemos en esta pantalla se han usado 4 a´s como contraseña por lo que el resultado que se obtiene, al ser una operación nula como es el XOR, es el texto en claro ocultado en la imagen como se puede observar en la siguiente captura de pantalla:



Una vez establecidas estas hipótesis, se intentó descubrir cual era el proceso real que seguía el sistema para cifrar la información. Es un sistema que no introduce la contraseña utilizada en la imagen, si no que para conocer el contenido de un texto cifrado, es necesario saber cual es la contraseña que se ha utilizado. Conocida ésta y conocido el sistema que utiliza PGE v1.0 para cifrar la información, que pasaremos a comentar a continuación, seremos capaces de conocer el mensaje oculto en cualquier fotografía tratada con este programa.

Después de realizar diversas pruebas, siempre siguiendo la misma filosofía que hasta ahora, se ha detectado que el sistema trabaja de dos formas distintas, dependiendo de la longitud de la clave proporcionada por el usuario. Los dos casos que se dan son:

- 1 carácter en la clave.
- Más de 1 carácter en la clave.

Para el primer caso, 1 carácter en la clave, el sistema realiza una operación XOR entre el texto a ocultar y un código correspondiente al carácter introducido en la clave. La lista de códigos que utiliza PGE v1.0 se ha obtenido haciendo diversas pruebas con cada uno de los posibles caracteres, obteniendo como resultado la siguiente tabla:

Carácter	código hexadecimal en pge v1.0
a	0801
b	101010101084
c	181818181818181818181809
d	20202090
e	2828282828282828282828282819
f	303030303030303030A4
g	3838383838383831
h	40404040404040C0
i	48484848484848484851
j	505050505050505050505050E4
k	58585879
l	606060606060606060606010
m	6868686868A9
n	7044
ñ	20202090
o	78
p	80
q	8821
r	9090909090C4
s	98989898989898989989869
t	A0A0A010
u	A8
v	B0B0B0B0B0B0B0B0B064
w	B8B8B8B8B8B8B8B811
x	C0
y	C8C8C8C8C8C8C8C871
z	D0D0D0D0D0D0D0D0D0D0D0D0D024
Blanco	00
0	80
1	88A1
2	9090909090C4
3	98989898989898989898E9
4	A0A0A010
5	A8A8A8A8A8A8A8A8A8A8A8A839
6	B0B0B0B0B0B0B0B0B064
7	B8B8B8B8B8B8B8B891
8	C0

Carácter	código hexadecimal en pge v1.0
9	C8C8C8C8C8C8C8C8F1
!	0881
“	101010101084
#	181818181818181818181889

$	20202090
%	28282828282828282828282899
&	303030303030303030A4
‘	3838383838383838B1
(	40404040404040C0
)	484848484848484848D1
*	505050505050505050505050E4
+	585858F9
,	606060606060606060606010
-	686868686829
.	7044
/	78
:	D0D0D0D0D0D0D0D0D0D0D0D0D024
;	D8D8D859
<	E0E0E0E0E0E0E0E0E0E0E090
=	E8E8E8E8E8E8E8E8C9
>	F004
?	F8
@	00
[	D8D8D8D9
\	E0E0E0E0E0E0E0E0E0E090
]	E8E8E8E8E849
^	F004
_	F8
`	00
{	D8D8D8D9
|	E0E0E0E0E0E0E0E0E0E090
}	E8E8E8E8E849
~	F004
¿	40404040404040C0
¡	686868686829
ç	38383838383838B1

Para que quede un poco más claro, se procederá a mostrar un ejemplo:

1. Texto a ocultar = “CarlosIII”, en hexadecimal “4361726C6F73494949”
2. Clave “r“
3. r en PGE v1.0 de acuerdo con la tabla = 9090909090C4
4. Operaciones:
   i.

      4361726C6F73494949

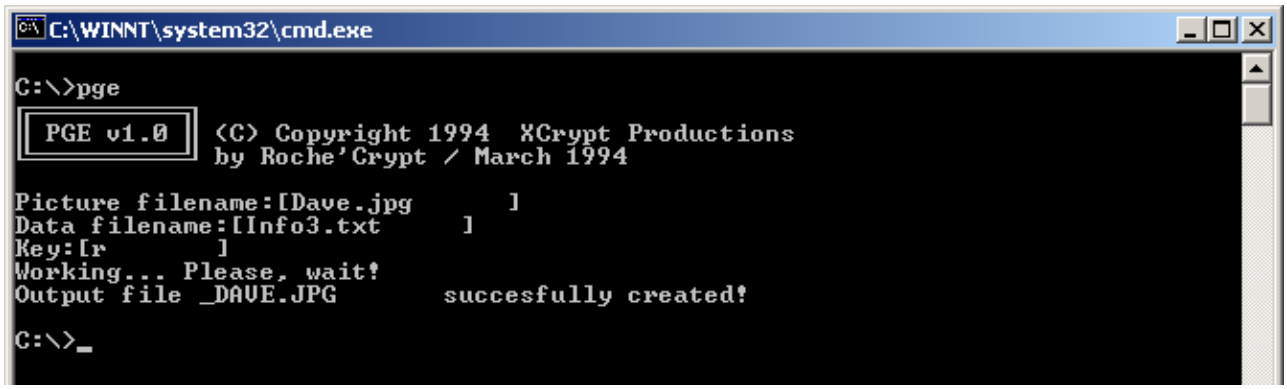
      XOR

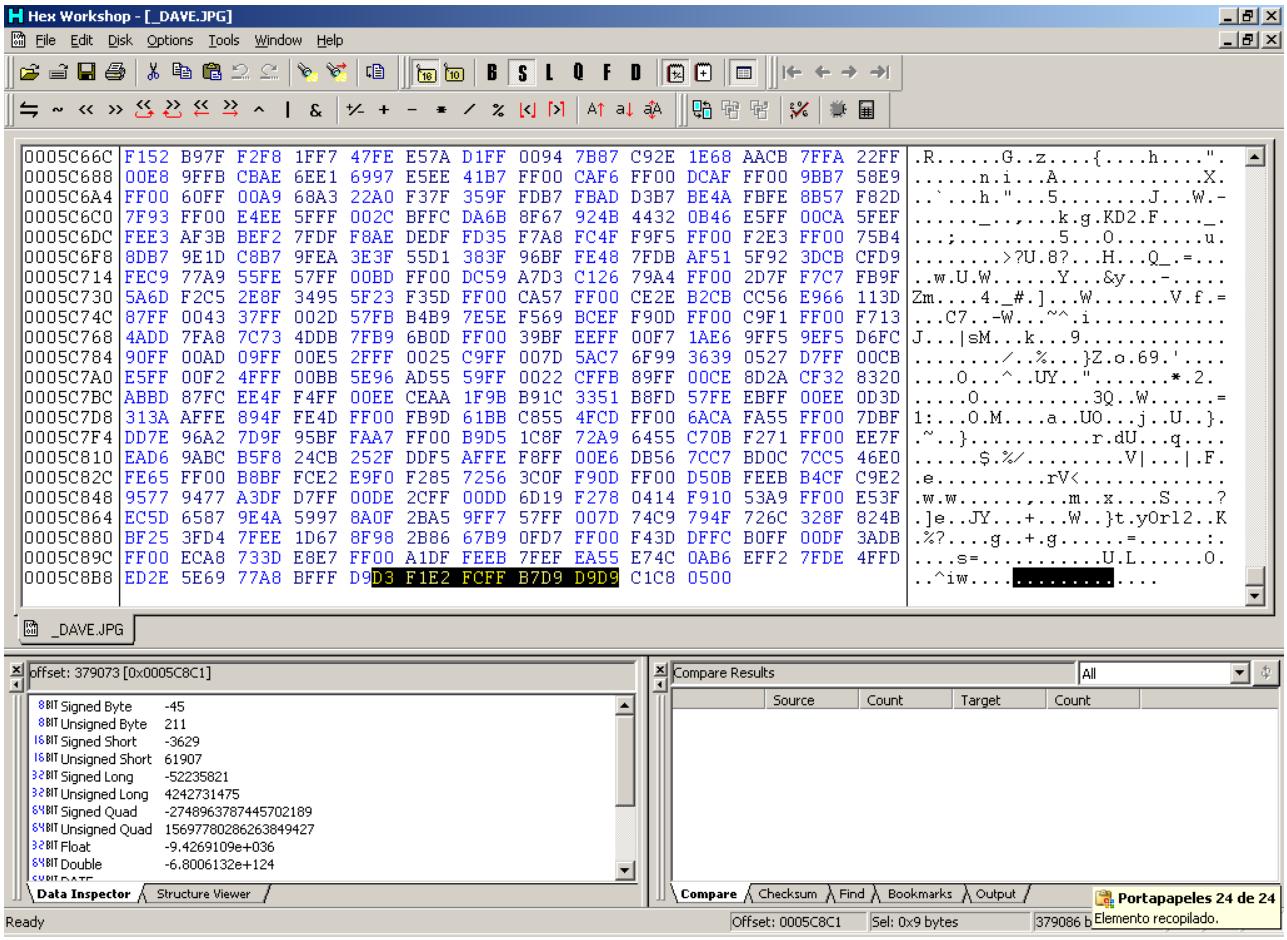
      9090909090C4909090 (se repite el código cuantas veces sea necesario de forma cíclica)

D3F1E2FCFFB7D9D9D9

5. Resultado = D3F1E2FCFFB7D9D9D9

Comprobemos estas operaciones con los resultados que nos devolvería PGE v1.0 a ver si coinciden:





Como se puede observar en la imagen anterior, el resultado obtenido a través de los cálculos realizados coincide plenamente con el proporcionado por PGE v1.0 corroborando el estudio realizado.

Para el segundo caso, Más de 1 carácter en la clave, el sistema realiza operaciones XOR con cada uno de los primeros pares hexadecimal correspondiente al código de cada uno de los caracteres de la contraseña proporcionada para cifrar y el texto a ocultar. Por ejemplo, si el texto que queremos cifrar es "CarlosIII" y la contraseña es "1b.-" el sistema hará lo siguiente:

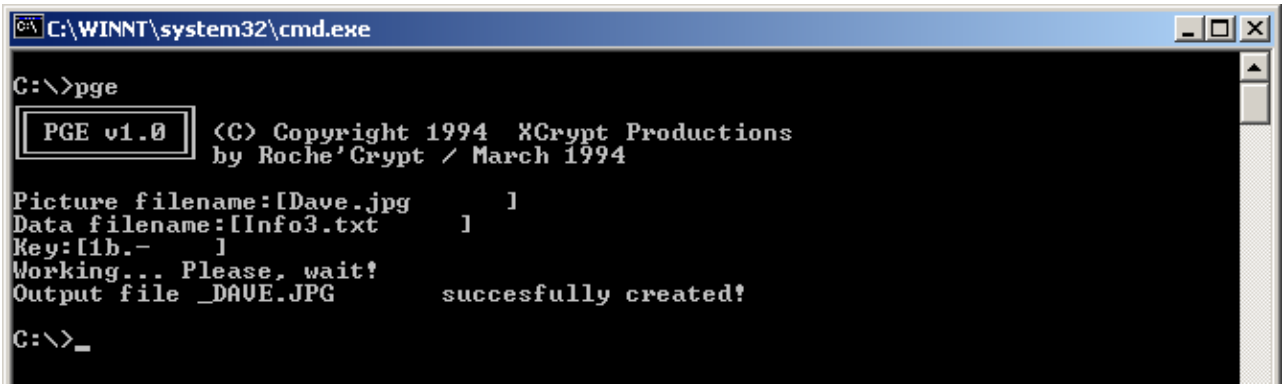
1. 1º par Hex "1" XOR 1º par Hex "b" XOR 1º par Hex "." XOR 1º par Hex "-" = ClavePGE
2. "CarlosIII" (en Hexadecimal) XOR ClavePGE (se repite el código cuantas veces sea necesario de forma cíclica) = Resultado
3. Resultado sería el contenido final que aparecería oculto en la imagen.

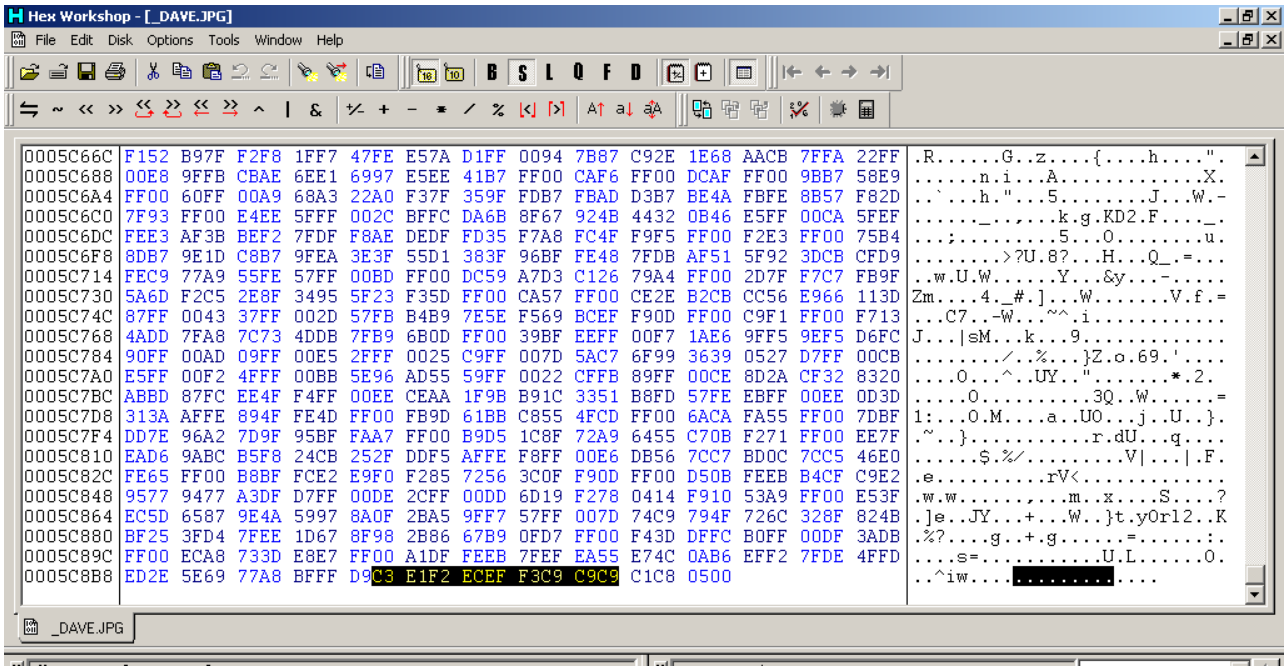
Como se ha indicado en el caso anterior, un solo carácter en la clave, el sistema utiliza unos códigos específicos para cada uno de los caracteres posibles de la contraseña y con ellos realizar la operación XOR indicada.

Un ejemplo para clarificar este caso:

1. Texto a ocultar = "CarlosIII", en hexadecimal "4361726C6F73494949"
2. Clave "1b.-"
3. 1 en PGE v1.0 = 88A1
4. b en PGE v1.0 = 101010101084
5. . en PGE v1.0 = 7044
6. - en PGE v1.0 = 686868686829
7. 88 XOR 10 XOR 70 XOR 68 = 80
8. Operaciones:
    i.
        4361726C6F73494949
        XOR
        8080808080808080 (se repite el código obtenido tantas veces como sea necesario)
        C3E1F2ECEFF3C9C9C9
9. Resultado final = C3E1F2ECEFF3C9C9C9

Comprobemos estas operaciones con los resultados que nos devolvería PGE v1.0 a ver si coinciden:





Como se puede observar en la imagen anterior, el resultado obtenido a través de los cálculos realizados coincide plenamente con el proporcionado por PGE v1.0 corroborando el estudio realizado.

5.Forma de uso en UTC

Ha sido uncluido como un nuevo módulo de JPEG's.

6.Información adicional

Es importante comentar, que para la realización de las pruebas preliminares se ha hecho uso de una fotografía JPG normal y corriente, y de un fichero de texto cuyo contenido era tan solo “Hola”, que en hexadecimal tiene el siguiente aspecto: “486F 6C61”.

7.Documentación adicional

www.jjtc.com/Security/stegtools.htm

www.surfsecret.com/products/snews/issues/121903.html

SECCIÓN 2.2
INSERTANDO DATOS EN CABECERA

DOCUMENTACIÓN MÓDULO INVISIBLE SECRETS 2002

1.Explicación del módulo

Invisible Secrets es un shareware que fue creado por NeoByte Solutions, la versión de prueba vale 30 días , tiene una interfaz bastante buena y con muchas opciones pero a nosotros sólo nos interesa la parte de detección de Esteganografía que no resulta demasiado complicada por lo que no es un método bueno para ocultar información realmente relevante.

Interfaz de Invisible Secrets:

2.Motivos para hacerlo

Este software tiene muchas mejoras con respecto a los que sólo se limitaban a añadir los datos al final del fichero y eso dará a UTC un mayor abanico de posibilidades a la hora de detectar imágenes ocultas .

3.Especificación

No sólo soporta ficheros JPEG, sino también PNG,BMP,HTML y WAV) y además es un software abierto a incluir otras extensiones incluyendo también sus algoritmos criptográficos correspondientes, y eso es lo que buscamos en UTC, una buena modularización que nos permita que el software siga creciendo, ya que la esteganografía es una técnica que sigue desarrollandose en función de las nuevas necesidades.

4.Desarrollo del módulo

Según los tipos de ficheros que trate funciona de diferente manera:

1.Para JPG's o BMP's los datos ocultos se colocan en el campo comentario ("comment field") , que está situado al principio del fichero en el caso de los JPG y al final , en el caso de los PNG.

2.Y en el caso de los BMP y WAV usan el método del "bit menos significativo",aunque existe un agujero de seguridad de este software para BMP's es que los LSB'S (bits menos significativos) no son usados por los datos escondidos, se limita a poner todos a 1 o a 0.

3.En el caso de los HTML usa espacios o tabuladores añadidos al final de las líneas.

5.Forma de uso en UTC

Hemos implementado un módulo de detección de imágenes ocultas pero en el caso de este software la posibilidad de añadir nuevas funcionalidades es muy grande, porque al soportar diferentes extensiones y diferentes algoritmos de encriptación, la implementación de nuevas detecciones con "Invisible Secrets" es muy grande y llevaría mucho tiempo , se podría considerar como una aplicación independiente a UTC.

6.Informes adicionales

Las mejoras que introduce este software son:

1.Uso de algoritmos de criptografía (Blowfish,Twofish,RC4,AES) y la posibilidad de más tarde usar nuevos métodos criptográficos que veamos necesarios añadiéndolos como librerías.

2. Comprime los datos antes de esconderlos de forma que reduce el tamaño y la renundancia de los datos ocultos.

3.Permite añadir ficheros "fake"(ficheros random) para incrementar el ruido.

Otros aspectos negativos de este software es que no es fuerte a posibles ataques porque la forma de ocultar la información oculta , por ejemplo en el campo comentario de los JPG y PNG es muy fácil de detectar y lo mismo para el resto de extensiones.

Y lo oculta siempre de la misma manera, aunque los datos luego tengan una fuerte encriptación es bastante sencillo ver que hay información escondida porque la manera es siempre la misma.

7.Documentación adicional

http://invisible-secrets.uptodown.com/

www.neobytesolutions.com/ -

http://www.softpedia.com/public/cat/14/1/14-1-20.shtml

http://www.coxsoft.com/system-utilities/invisible-secrets.htm

SECCIÓN 2.3
EMBEBIENDO DATOS FIJOS

DOCUMENTACIÓN MÓDULO INPLAINVIEW

Lista de Contenidos

1.Explicación del módulo

Se trata de un software que oculta texto en una imagen BMP modificando los bits menos significativos(LSB) de cada uno de los pixeles.Se trata de un software que nos da más seguridad a la hora de ocultar información de las que nos puede dar Camouflage o JpegX, pero sigue siendo de fácil detección.

2.Motivos para hacerlo

Se trata de un software bastante interesante en su funcionamiento desde nuestro punto de vista y que incluye ocultación de imágenes en BMP,extensión que hasta ahora no habíamos incluido y que es bastante usada en la esteganografía.

3.Especificación

Trabaja con extensiones BMP y de ellos tenemos que conseguir los LSB's de cada pixel, ya que InPlainView consiste en modificar estos bits.

4.Desarrollo del módulo

En un fichero BMP de 24 bits no existe paleta de colores, para cada píxel tenemos un byte para el color azul saturado,de manera que tenemos 256 valores posibles y lo mismo para el verde y el rojo(RGB),cada píxel necesita por lo tanto 3x8=24bits , lo que significa que podemos tener un número de valores igual a 256x256x256,lo que se traduce en aproximadamente 16 millones.

Si modificamos el LSB de cada píxel(incrementándolo o aumentándolo) no se cambiará el valor de cada uno de los valores RGB en más de 1/256 lo que no es visible por el ojo humano.

Pero con este cambio no es suficiente porque sería algo demasiado evidente para detectar por un software.Lo que hace InPlainView es esconder la cadena de bits empezando por el primer píxel y hasta el siguiente hasta que no tiene nada más que esconder.Esto crea un fácil patrón a reconocer al principio del fichero ,incluso si el texto escondido está totalmente encriptado, por lo que es fácil detectar la presencia de datos ocultos, aunque luego no sea fácil la recuperación de ellos.

Diversos estudios demuestran que el concepto de los LSB’s son procesos fijos y su colocación viene determinada por algún algoritmo, pero no es así, están colocados de manera aleatoria.Nosotros hemos implementado este módulo eliminando todos los bits que forman la imagen de los 24 Bits(8x3) quedándonos sólo con los bits menos significativos(LSB).

Aquí hay un ejemplo de una imagen de Audrey Hepburn sacada de internet que contiene imágenes ocultas y un pequeño texto cuya longitud un pequeño bit y sacamos sus bits menos significativos para ver que no existe casi diferencia a la hora de sacar los LSB.

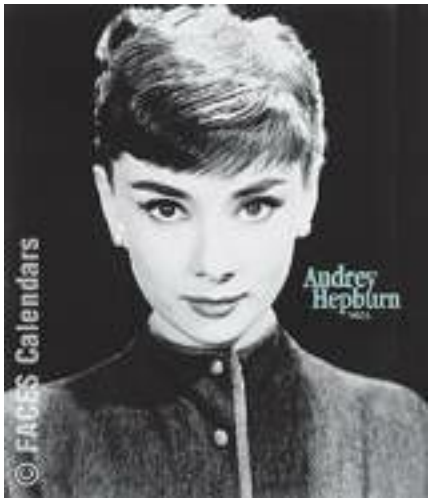


Imagen on imagen ocultaque esconde un pequeño texto

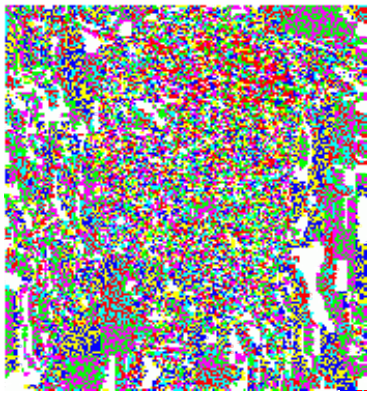

Imagen de los LSB de la imagen anterior sin texto.


Imagen de los LSB de la misma imagen con texto escondido.

Como se puede ver , existe gran diferencia entre la segunda y la tercera imagen viendo en la última al final de la imagen el texto escondido , por tanto modificar los bits menos significativos no es palpable por el ojo humano pero con software es fácilmente detectable la diferencia.Esta diferencia es detectable a través de funciones estadísticas del array de LSB's.

Ponemos otro ejemplo con una Audery Hepburn sin imagen oculta, para ver que el resultado es el mismo:


Imagen2 sin información oculta


Misma imagen2 viendo sólo los LSB.

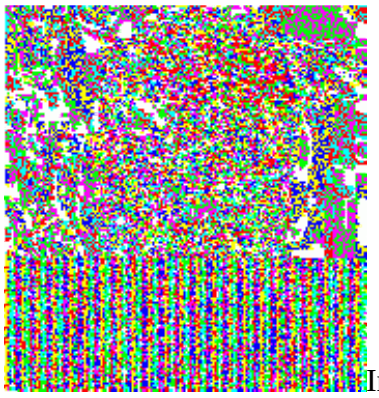


Imagen 2 de los LSB con el texto escondido.

Otra vez se puede apreciar claramente la existencia al final de la última imagen del texto oculto.

5.Forma de uso en UTC

A partir de ahora hemos incluido una nueva clase para tratamiento de ficheros BMP que nos da los datos de los pixeles para cada color en arrays y arrays con los bits menos significativos y métodos necesarios para su funcionamiento y llamada desde otros módulos.

6.Información adicional

Los buenos softwares de esteganografía se preocupan de mezclar los datos escondidos con el fichero de manera que nosea diferenciable usando funciones estadísticas, pero este no es el caso de InPlainView.

InPlainView funciona de manera que justo antes del texto oculto escribe 5 bytes, que pueden ser una especia de marca, lo que no es muy buena idea ,porque lo hace un software más fácilmente detectable porque a un atacante puede usarlo para buscar la presencia o no de datos.

Esta firma consiste en :

-2 bytes a 0.

-2 bytes que contienen el tamaño del texto escondido.

-1 byte que contiene 0 si la password no es usada y 1 en el caso contrario.

La encriptación es un simple XOR de la password con el texto.La longitud de esta encriptación depende del número de veces que repitamos la password y puede ser muy corta o extremadamente larga , lo que actualmente depende principalmente de dos variables:

-El tamaño de la password comparado con el tamaño del texto.

-La aleatoriedad del texto y la password porque los textos normales tienen patrones muy fácilmente reconocibles.

Si el tamaño de la password coincidiera con el tamaño del texto la encriptación sería prácticamente como la del "One Time Pad" que es un método encriptográfico totalmente detectable.Y en el caso de una password muy corta y un texto muy largo también sería fácil obtenerla.Este software se hace más seguro cuando usamos passwords bastante largas que no coincidan con el texto, lo que no suele ser muy frecuente, porque la mayoría de la gente suele usar passwords cortas y simples confiando en la seguridad del programa que en Internet su creador cataloga de "Media" cuando en realidad no llega a ser media, se queda tan sólo en baja a cualquier ataque.

El mecanismo para encontrar si hay imágenes ocultas se puede resumir en los siguientes pasos:

-Si el fichero no tiene 24 bits no nos vale, porque InPlainView sólo funciona con BMP's de 24 bits.

-Se localizan los primeros 5 bytes de los datos ocultos y se compara con el patrón que usa y si no es este patrón , el fichero no usa InPlainView.

-Miramos si el texto escondido no está encriptado y si lo está lo analizamos con funciones estadísticas.

7.Documentación adicional

www.softpile.com/Utilities/ Encryption/Review_05300_index.html

www.securityfocus.com/tools/1167/scoreit

inplainview.netfirms.com/

www.bhitomorrow.com/cat/352069

DOCUMENTACIÓN MÓDULO JSTEG

Lista de Contenidos

1.Explicación del módulo

Es el programa que más se usa en Esteganografía para formatos JPG aunque existan muchos otros para este tipo de ficheros.Antes hemos implementado otros módulos que tan sólo añadían datos detrás de un fichero que era visible y no necesitabamos más datos del fichero.Ahora necesitaremos la librería de los JPEG que nos permita acceder a los DCT en forma de arrays para poder trabajar con ellos.

2.Motivos para hacerlo

Aunque no dispone de encriptación es un método muy usado para embeber datos en ficheros JPEG porque añade más seguridad que los métodos que hemos explicado anteriormente en la sección 2.1.Y si es un método muy usado , nos ha parecido muy recomendable incluirlo en UTC , cuya principal finalidad es encontrar el máximo número de estegos cuando lancemos en la red nuestra aplicación.

3.Especificación

El código de Jsteg no es sencillo y necesitamos algunos datos de entrada que nos dén información sobre los bits del JPEG.Los ficheros JPEG usan diferentes espacios de colores que los bitmap.Los tres coeficientes de colores (Red,Green,Blue) RGB es un nuevo esquema basado en que cada pixel tenga tres valores que pueden tomar 256 posibilidades de color y también usan dos coeficientes definidos por Blue/Yellow y Red/Green, lo que llamamos valores YCbCr.

En Jsteg vamos a quedarnos con los valores YCbCr y a partir de ellos obtener los valores DCT.

4.Desarrollo del módulo

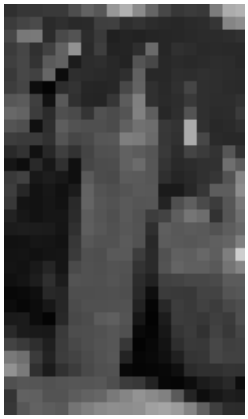
Debido a que el ojo humano es mucho más sensitivo a la luminosidad que a los colores, podemos borrar mucha información de color de las imágenes sin cambiar realmente la percepción que nosotros teníamos de la imagen,la mayoría de las veces los 4 pixeles adyacentes pueden ser más o menos reducidos a un solo valor , de esta manera podemos eliminar un número de información igual a ((1/3*3/4)*2) de la información de la imagen, lo que se traduce a la mitad, y es en este momento cuando empiezan a adquirir importancia los valores YCbCr.

Aquí ponemos un ejemplo de una imagen reducida a los valores YCbCr con un factor de 8, lo que significa que la información del color se ha hecho por bloques de 8x8 pixels.

Esta es la misma imagen con la intensidad Y:

Imagen con intensidad Cb(blue/yellow):

Imagen con intensidad Cr(red/green):



Ahora a los valores que nos quedan le aplicamos la DCT(Discrete Cosine Transform), que nos va a transformar los valores en frecuencias, lo que se hace es sacar dos coeficients a y b de la fórmula y=ax+b y se mira en qué coordenadas se acumulan más puntos de la imagen de manera que se pueda trazar una recta.

Se traducen los 64 valore en 64 coeficientes de frecuencia , la ventaja de tener coeficientes en lugar de valores es que los coeficientes son bajos y permiten reconstruir la imagen original con bastante semejanza.

Así es como más o menos quedaría la transformación:

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Valor de los Pixeles

1260	-1	-12	-5	2	-2	-3	1
-23	-17	-6	-3	-3	0	0	1
-11	-9	-2	2	0	-1	-1	0
-7	-2	0	1	1	0	0	0
-1	-1	1	2	0	-1	1	1
2	0	2	0	-1	1	1	-1
-1	0	0	-1	0	2	1	-1
-3	2	-4	-2	2	1	-1	0

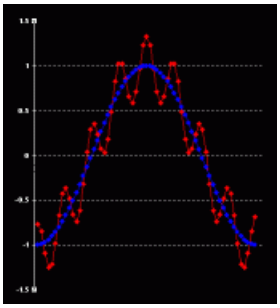
Valor de los DCT

Se puede apreciar la diferencia en los valores, los DCT son valores mucho más pequeños y que nos dan casi la misma información que la imagen original para buscar estegos o textos ocultos.

Jsteg trabaja con estos valores finales.Lo que hace es cuantificar los valores ,es decir, cada uno de los coeficientes se comparan con valores fijos de una tabla , viendo cuál es el valor más alto y eliminándolo , para eliminar los valores de alta frecuencia.Estas tablas fijas vienen dadas por el tanto por ciento de calidad con el que guardas un fichero JPEG.

La mayoría de los valores que nos quedan son cero y es en estos bits donde Jsteg esconde los datos.

Esta es una gráfica que muestra la línea de la alta frecuencia(la roja) y la baja frecuencia (la azul):



Nosotros al cuantificar nos estamos quedando sólo con la azul, esto significa que nos estamos quedando con la información principal que viene dada por la baja frecuencia y estamos eliminando los pequeños detalles que vienen dados por la alta frecuencia(línea roja).

La manera de detectar un fichero que tiene imágenes camufladas con el método de Jsteg es a través de la función estadística chi-cuadrado que hemos incluido en el "ToolKit" de nuestra aplicación y que detecta la información oculta que está embebida de manera secuencial.Aplicando dicha función sobre los valores DCT podemos obtener la presencia o no de estegos.

La función chi-cuadrado también se usa para detectar la presencia de estegos en los bits embebidos de manera secuencial.

5.Forma de uso en UTC

En UTC la implementación de Jsteg fue el primer módulo en el que usamos los valores DCT de un BMP, que ya teníamos implementado en la librería de uso de este tipo de ficheros.La librería después de decodificar un bloque de coeficientes DCT , llama a un rutina para colocarlos, en vez de en zig-zag, de manera natural.

En nuestro programa sustituimos esta llamada por una llamada a nuestro programa.Nosotros sólo necesitamos calcular donde tendría que ir esa llamada en nuestro módulo , salvar la llamada original y reemplazarla por el nuevo valor.

Una vez que trabajamos en nuestro módulo , transferimos los valores de los coeficientes DCT desde mi librería a memoria y saltamos donde se supone que estaba la llamada original a esa rutina y ya tenemos todos los coeficientes DCT cuantificados y eso es todo lo que necesito porque los datos emebebidos por Jsteg están en esos coeficientes.

6.Informes adicionales

JSteg maneja el formato más complejo de lso JPG y emebebe la información escondida de una manera bastante simple, que incluso viene detallada en el readme.La información escondida no está ni siquiera encriptada.Y la recuperación se basaría en la forma de extraer los datos de los coeficientes DCT.

Básicamente los coeficientes DCT cuyo valor es 0 no se modifican, y los otros son usados para embeber secuencialemente un bit de la información oculta sobrescribiendo los LSB's.La información oculta tiene este formato:

A:Son 5 bits que expresan la longitud en bits del campo B.

B: son un número de bits desde cero a 31 que expresan la longitud en bits del fichero que se esconde.

C:son los bit del fichero escondido.

De esta forma nosotros extrayendo un bit de cada LSB, vemos que de los primeros 5 coeficientes DCT obtenemos el tamaño del campo A, y obtenemos el número de bits que tienen los datos ocultos(campo B) y finalmente podemos extraer los datos ocultos(campo C) reconstruyendo los bytes desde los bits que hemos obtenido.

Esto es lo que tendríamos que implementar en el módulo de recuperación de datos que nosotros no hemos implementado aún, y por ahora nos hemos limitado tan sólo obtener la información de si hay o no estego.

Este es un ejemplo de una imagen con y sin información ocultada a través de Jsteg:

Imagen1 original sin datos ocultos ocupa 16778 bytes:

Imagen2 con datos ocultos,un poema,ocupa 17344 bytes:

Estos son los coeficientes DCT de la imagen de arriba sacados en bloques de 8X8(64 bytes) de la Imagen 1:

D6 69 13 05 03 15 F2 EB

FF 04 01 00 FA FB F9 FF

06 02 FE FF 00 00 00 FF

01 03 02 01 01 FF 00 00

01 00 00 00 00 00 00 00

00 FF FF 00 00 00 00 00

00 00 00 00 00 01 00 00

00 00 00 00 00 00 00 00

Ahora vemos los coeficients DCT del mismo bloque después de esconder el texto, serían los coeficientes correspondientes a la imagen 2:

D6 69 12 05 03 15 F3 EA

FE 04 01 00 FA FB F8 FE

06 03 FE FF 00 00 00 FE

01 02 03 01 01 FE 00 00

01 00 00 00 00 00 00 00

00 FE FF 00 00 00 00 00

00 00 00 00 00 01 00 00

00 00 00 00 00 00 00 00

Y ahora vemos el mismo bloque pero sólo con los LSB de cada byte:

00 01 00 01 01 01 01 00

00 00 01 00 00 01 00 00

00 01 **00 01** 00 00 00 **00**

01 **00 01** 01 01 **00** 00 00

01 00 00 00 00 00 00 00

00 **00 01** 00 00 00 00 00

00 00 00 00 00 01 00 00

00 00 00 00 00 00 00 00

Aparecen subrayados los LSB modificados por Jsteg, Jsteg modifica sólo aquellos LSB que no tienen el valor correcto.En cada bloque los primeros 5 campos de bits(rojo) son el campo A, en verde el B y en negrita el campo C que equivale a la información oculta por Jsteg.

7.Documentación adicional

www.jjtc.com/stegoarchive/stego/software.html

portal.acm.org/citation.cfm?id=589936

www.cs.dartmouth.edu/~farid/publications/tr01.pdf

DOCUMENTACIÓN MÓDULO IMAGEHIDE

Lista de Contenidos

1.Explicación del módulo

Se trata de un software de esteganografía cuya versión original está implementada en Delphi y es de libre distribución .Consiste en encriptar texto y esconderlo en formatos de imágenes, y funciona normalmente con ficheros de tipo BMP aunque se podría implementar para nuevas extensiones.

2.Motivos para hacerlo

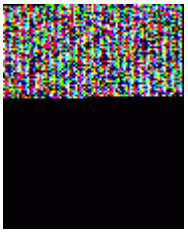
Aunque su autor dice claramente en su página web que el programa sólo se encarga de encriptar el texto, nos ha parecido interesante ver como maneja los ficheros BMP para insertar texto detrás sin que sea visible por el usuario, además añade más métodos esteganográficos que trabajan con extensiones BMP a nuestra aplicación.

3.Especificación

Se usa para encriptar ficheros de texto en imágenes BMP modificando los LSB's por lo que necesitamos obtener los coeficientes DCT que tenemos como salida de datos en nuestra librería de tratamiento de ficheros BMP.

4.Desarrollo del módulo

ImageHide trata de esconder un texto en un fichero de 24 bits bitmap y después sacar los LSB(Least Significant Bits) , de manera que nos queda algo como esto:



Los bits escondidos no se han escrito de manera aleatoria por toda la imagen , sino que están escondidos linealmente empezando por el final del bitmap, que es la última línea en el fichero.

Para extraer los datos ocultos nosotros sólo tenemos que extraer los “Least Significant Bits” , transformarlos en bytes y mirar la información extraída.Nosotros veremos que se encuentran en el siguiente formato:

-Los primeros 4 bytes (32 bits) están puestos a 0.

-Los siguientes 4 bytes(32 bits) indican el tamaño del texto que está escondido tras la imagen.

-Los siguientes bytes son los datos escondidos.

Por lo tanto , nosotros podemos usar los primeros 32 LSB’s que son fijos,(siempre a 0) como una marca que buscaremos para detectar la presencia o no de estegos.Si ellos son todos 0 es muy probable que la imagen pueda contener datos ocultos con ImageHide, aunque se pueden producir falsos positivos con un tipo especial de imágenes, por ejemplo los BMP que sean completamente negros.

5.Forma de uso en UTC

Para implemetar dicho módulo hemos tenido que usar de nuevo nuestra librería de tratamiento de ficheros BMP que hemos adaptado para nuestra aplicación, porque necesitamos de nuevo la información de los LSB’s para ver si existe o no información oculta.

6.Informes adicionales

En caso de que una vez usamos UTC detectamos que el mensaje está encriptado, no es muy difícil de desencriptar , ya que el algoritmo de encriptación que usa es simple.Básicamente consiste en un bucle XOR que se aplica byte a byte con una clave que es modificada en cada paso, mediante dos sumas y una multiplicación).

Con cada uno de los bytes también hace una ordenación usando el byte que ha codificado antes.Nosotros no hemos llevado a cabo la implementación , pero no parece un algoritmo muy difícil de romper.

7.Documentos adicionales

www.computeractive.co.uk/downloads/1153818

www.sofotex.com/ImageHide-download_L16436.html

www.nonags.com/nonags/security.html

SECCIÓN 2.4
EMBEBIENDO DATOS ALEATORIOS

DOCUMENTACIÓN MÓDULO JPHIDE

Lista de Contenidos

1.Explicación del módulo

JpHide y JpSeek son programas que permiten esconder un fichero en otro fichero JPEG, es un método esteganográfico bastante interesante aunque no es difícil su detección pero sí su recuperación, ya que los datos los tiene encriptados mediante Blowfish, de manera que muchas veces es imposible probar que existe una imagen oculta.

Fichero sin información oculta:

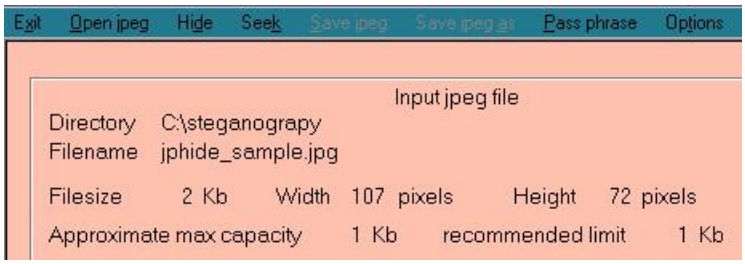
Fichero con información oculta:

2.Motivos para hacerlo

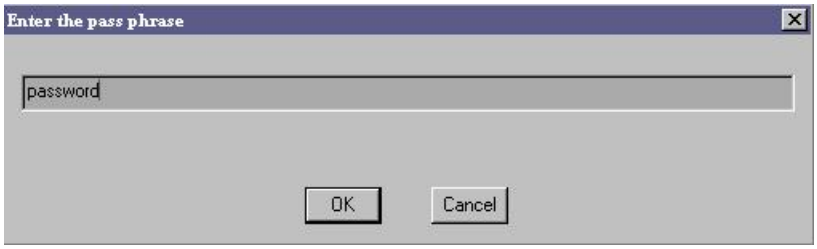
Aunque no sea un algoritmo de los más seguros es bastante usado en la red, y en UTC buscamos encontrar el máximo de estegos posibles.

3.Especificación

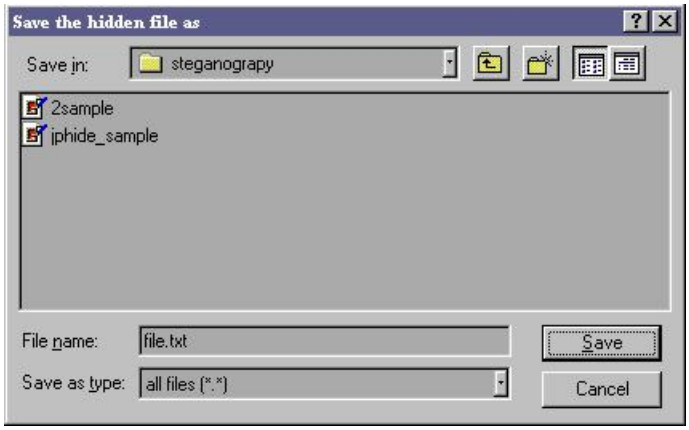
Cuando encontramos una imagen que creemos que contiene imagen oculta guardamos la imagen en nuestro disco duro y lanzamos el Jpeg.Tendremos lo siguiente:



Daremos a "Seek" y nos saldrá la siguiente pantalla:



Insertamos la password y nos aparecerá la pantalla con el fichero escondido:



El fichero se puede guardar con un nombre o con una extensión diferente a la original si así lo queremos.

4.Desarrollo del módulo

JPHide usa el algoritmo Blowfish para conseguir una cadena de bits que han sido colocados de manera aleatoria que determinan dónde se han almacenado los bits del fichero oculto.

5.Forma de uso en UTC

Nosotros no hemos implementado la parte de recuperación una vez detectados los datos, se podría incorporar esta parte más tarde como un módulo aparte.

6. Información adicional

Usando ficheros JPEG de unos 200Kb podemos insertar hasta una imagen de 20Kb con el mínimo efecto visual y estadístico.Hasta 35Kb es posible que los efectos a nivel visuales y estadísticos sean visibles y el programa rechazará insertar datos de más tamaño porque la presencia de información oculta sería obvia.

7.Documentación adicional

www.citi.umich.edu/u/provos/stego/usenet.php

www.mirrors.wiretapped.net/security/ steganography/jphs/jphs-README.txt

www.snapfiles.com/opinions/for_jphide.html

SECCIÓN 2.5
EMBEBIENDO DATOS ALEATORIOS Y FUNCIONES ESTADÍSTICAS

DOCUMENTACIÓN MÓDULO OUTGUESS

Lista de Contenidos

1.Explicación del módulo

Outguess es una herramienta esteganográfica que permite la inserción de datos ocultos en los bits renundantes de los datos de origen.La naturaleza de los datos es irrelevante para Outguess.

Se puede obtener como un tar de UNIX y está disponible como un software de licencia BSD y es completamente gratis.

2.Motivos para hacerlo

Outguess es una herramienta de esteganografía bastante potente que ya incluye un nivel de seguridad importante tanto en detección como en recuperación.Para UTC es un programa muy útil porque permite manejar nuevas extensiones de ficheros siempre se que sea capaz de proveer al programa con la librería de cada una de las extensiones, y por

tanto el desarrollo de nuevos módulos es muy factible, y eso es lo que buscamos con nuestra aplicación.

3.Especificación

Según las versiones del Outguess soporta unas extensiones distintas, para la versión 0.2 se pueden usar ficheros PNM y JPEG pero permite añadir más tipos de datos siempre que la librería de uso de esos datos sea añadida.

No es útil la función estadística chi-cuadrado porque aunque embebe datos al igual que lo hacen Jsteg o JpHide modifica los LSB y la función chi-cuadrado , en ese caso ya no es efectiva.

La forma de funcionamiento de Outguess se puede resumir en los siguientes puntos:

1. Lo primero que hace el Outguess es embeber los bits de manera aleatoria dejando los LSB a valor 1 o a valor
2. Después de embebida la imagen es procesada otra vez y se modifican los bits necesarios para que el histograma de la imagen digital escondida(el estego) coincida con el histograma de la imagen digital visible, por esto al función chi-cuadrado no puede detectar la presencia de imágenes ocultas con Outguess porque el ataque a la función chi-cuadrado se basa en analizar las estadísticas de primer orden del estego.

La posible forma de detectar la presencia de datos que su creador Provos quería evitar está basada en el hecho de que el mecanismo de embeber datos con Outguess se basa en sobreescribir lso LSB's.Esto significa que embebiendo otro mensaje en el estego la idea de Provos pierde su efecto de manera que los histogramas de la imagen escondida y de la imagen que lo cubre ahora son diferentes.

4.Desarrollo del módulo

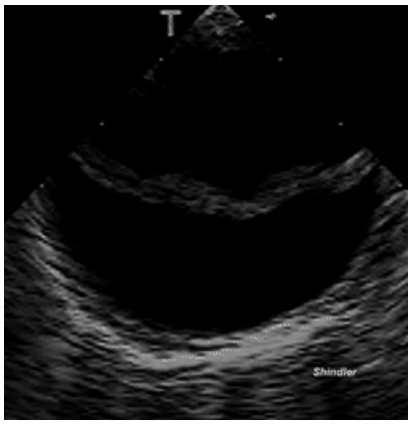
Para imágenes JPEG , Outguess usa funciones estadísticas basadas en contadores de frecuencias, estas funciones estadísticas no detectan la prensencia de imágenes ocultas y Outguess puede determinar el tamaño máximo del mensaje que puede ser escondido mientras es capaz de mantener las estadísticas basadas en contadores de frecuencias.

Outguess usa un objeto " generic iterator" para seleccionar los bits en los datos que deben ser modificados y son embebidos en los datos que contienen el resto del mensaje.

Estos son dos imágenes, una con información oculta a través del Outguess y la misma imagen sin estegos:

Original:

Con información oculta:



5.Forma de uso en UTC

Nosotros no hemos implementado las funciones estadísticas que ya están hechas multitud de veces por otros desarrolladores, en este caso hemos usado el StegDetect que detecta la presencia de imágenes ocultas para el Outguess.

6.Informes adicionales

Lo que tiene Outguess a favor si lo comparamos con otros métodos esteganográficos vistos anteriormente es:

1.No es posible su detección por una función estadísticas basada en contadores de frecuencia, lo que hace mucho más complicada su detección en el caso de guardar información oculta.

2.Determina una tamaño máximo de mensaje a partir del cual asegura que la información se oculta sin posibilidad de que posibles atacantes puedan detectarlo, aunque ya hemos visto que eso no es del todo cierto, pero al menos aporta más seguridad que otros programas.

3. En el caso de los ficheros JPG permite que los mensajes ocultos sean más grandes que con otros métodos esteganográficos que también soportan esta extensión.

7.Documentos adicionales

www.software-facilities.com/ security-software/outguess.php

http://librenix.com/?inode=3721

www.topology.org/soft/crypto.html

SECCIÓN 2.6
MÓDULOS ESTEGANÁLISIS EN TEXTO

DOCUMENTACIÓN MÓDULO SPAMMIMMIC

Lista de Contenidos

1.Explicación del módulo

Consiste en mandar un mensaje de texto encriptado en un mail en la parte del spam de forma que la persona que lo recibe desconoce la existencia de este mensaje oculto.

2.Motivos para hacerlo

Aunque la esteganografía en texto no es tan interesante, es algo que existe y que se usa en la red.Podrái sorprendernos si insertamos los textos típicos en que se transforma el spammic en un buscador de internet la cantidad de mensajes ocultos que encontramos.Por eso nos ha parecido bien meterlo dentro de nuestra aplicación.

3.Especificación

Spammimic se encarga de un mensaje corto ,encriptarlo y esconder en él otro mensaje que parece ser un spam.Pero la persona que recibe el mensaje puede cortar y pegar el spam y decodificarlo para obtener el mensaje original.Es bastante detectable y se usa como diversión.

4.Desarrollo del módudo

Se trata de un algoritmo de encriptación de text, tan sólo habrá que coger el texto que creeemos tiene el mensaje oculto y desencriptarlo.

5.Forma de uso en UTC

Lo hemos incluido como un módulo nuevo y busca los patrones en los que convierte el texto Spammimc.

6.Información adicional

Este programa puede encontrarse en la página web:

http://spammimic.com/

7.Documentos adicionales

spam.gunters.org/archive/2000/12/18/spammimic

amsterdam.nettime.org/Lists-Archives/ rohrpost-0104/msg00217.html

DOCUMENTACIÓN MÓDULO SNOW

Lista de Contenidos

1.Explicación del módulo

SNOW es un programa que permite ocultar información dentro de cualquier fichero de texto sin afectar a este en apariencia. Permite también la posterior extracción de la información oculta

2.Motivos para hacerlo

Incluir un nuevo módulo de esteganografía en texto.

3.Especificación

Codifica el mensaje que quiere ocultar mediante espacios y tabuladores (blancos) al final de las líneas del fichero portador, con lo que en apariencia no cambia. Con esto consigue que al observar el fichero de texto sigamos viendo la información que contenía originalmente.

4.Desarrollo del módulo

Tiene la posibilidad de tratar con anterioridad la información que se va a ocultar con dos técnicas:

- La primera técnica es la compresión de la información mediante el algoritmo de Huffman, con lo que se consigue reducir su tamaño con las consecuentes ventajas a la hora de ocultarlo, ya que el tamaño del fichero portador original y el fichero con la información oculta se diferencien lo mínimamente posible en tamaño (bytes).
- La segunda técnica que se puede emplear es el cifrado de la información mediante el algoritmo ICE (cifrador CFB) de 64 bits, diseñado por el autor de SNOW. Este cifrador utiliza un vector inicial que se autoencripta, cada vez que un bit es cifrado, el vector tiene un desplazamiento hacia la izquierda y el bit encriptado es añadido al vector por la derecha.

Estas dos técnicas no son excluyentes, es decir, SNOW permite realizar ambas a la vez sobre una misma antes de ocultarla en un fichero de texto.

Por último voy a describir el algoritmo de ocultación de información de SNOW:
Lo primero que hace es insertar una marca de comienzo a modo de cabecera de mensaje, que estará representada por un tabulador, y que se colocará al final de la primera línea en la que exista espacio para meter blancos. Posteriormente, para ocultar la información seguirá añadiendo grupos de espacios en blanco, estos grupos tendrán una longitud variable, entre 0 y 7 espacios y representan la codificación de 3 bits:
000 = 0,
001 = 1,
... ,
110 = 6,
111 = 7.

Para separar los grupos unos de otros se utilizaran los tabuladores, es decir, el tabulador actuará como marca de fin de grupo.

5.Forma de uso en UTC

Lo incluimos como un nuevo módulo de esteganografía en texto.

6.Información adicional

QSS (Quick See Spaces) es un programa que analiza ficheros de texto y decide si contienen información ocultada por SNOW. Para ello se ha realizado un estudio de las características del algoritmo de ocultación de SNOW y se ha intentado aprovechar alguna de sus debilidades.

Lo primero que busca QSS en un fichero de texto es la marca de comienzo, el tabulador, la cabecera de mensaje que siempre inserta SNOW en todos los ficheros portadores de la información oculta.

Esta búsqueda se realiza mediante la lectura secuencial (carácter a carácter) del fichero de texto. Si se llega al final del fichero sin encontrar la marca, QSS indicará que el fichero no contiene información oculta. Si por el contrario la encuentra en algún lugar del texto, el programa pasará a analizar los últimos caracteres de cada línea, siempre que estos sean blancos.

QSS buscara el patrón dejado por SNOW en estos caracteres, este patrón tiene las siguientes características: grupos de 0 a 7 espacios separados entre si por tabuladores, es decir, cada grupo de espacios terminara con un tabulador excepto el último de la línea. Para representar un grupo de 0 espacios Irán 2 tabuladores seguidos. En cuanto alguno de estos patrones no se cumpla se descartara el fichero como posible portador de información oculta.

Otra característica que busca QSS es que no tenga líneas de mas de 512 caracteres de largo, ya que SNOW no las permite, es decir, si SNOW esconde información en un fichero de texto en el que haya líneas mas largas de 512 caracteres, estas se dividirán en varias líneas de 512 caracteres cada una excepto la última que tendrá de longitud el modulo de la longitud original entre 512.

Si el programa ha encontrado una marca y un posible patrón de información oculta indicará que el fichero analizado contiene información oculta.

El modo de empleo del programa es el siguiente:

C:\QSS <fichero>

Siendo <fichero> el nombre del fichero que se desea analizar. Este fichero deberá estar en el mismo directorio que el ejecutable de QSS.

7.Documentación adicional

www.darkside.com.au/snow/

www.burks.de/stegano/snow.html

SECCIÓN 2.7
CÓMO HACER UN MÓDULO EN UTC

Lista de Contenidos

1.Requerimientos necesarios

Los módulos están en la carpeta “modules”. Típicamente, usamos 4 ficheros para implementar la detección de una nueva firma o de un programa de esteganografía:

- ModuleX.h: Define la interfaz del módulo.
- ModuleX.cpp: Implementa el proceso de detección de un programa de ocultación de contenidos sobre unos determinados medios.
- ResultX.h: Interfaz para los resultados de ModuleX.
- ResultX.cpp: Guarda los resultados de la detección de ModuleX, y es capaz de devolver un p-valor sobre la probabilidad de que el medio tenga contenidos ocultos.

2.Forma inclusión

2.1 Añadir un módulo nuevo

Los módulos están en la carpeta “modules”. Típicamente, usamos 4 ficheros para implementar la detección de una nueva firma o de un programa de esteganografía:

1. ModuleX.h: Define la interfaz del módulo.

   Normas a seguir:

- Incluye los siguientes encabezados: #include "ResultX.h", #include "Module.h" y #include "ModuleHandler.h".
- Hay que definir un nombre que identifique unívocamente a este módulo, por ejemplo: #define MODULEX “MODULEX”. Con este nombre se podrá habilitar o

deshabilitar la ejecución de este módulo desde la línea de comandos, utilizando el "switch" –MODULEX (deshabilitar) o +MODULEX (sólo se ejecutará este módulo).

- La nueva clase debe heredar de Module, por lo que debes definir los métodos que eran virtuales puros: Result * apply(Media * media) , Result * getResult() y char * to_string().
- Usa un atributo de instancia: ResultX * result (para almacenar el resultado de la detección); y otro de clase: static Camouflage theModule (para poder registrar este módulo en el ModuleHandler durante la inicialización de la aplicación).

Algunos consejos:

- Recuerda incluir las directivas de preprocesador ifndef/define/endif, para que no haya problemas de duplicidad de definiciones.
- Si es necesario un destructor, es mejor que lo declares "protected", porque si después haces una subclase de este módulo podrás manejar mejor esta operación. Es posible que también sea útil por el mismo motivo declarar como "protected" lo que en principio debiera ser "private".
- Si añades más métodos públicos, para poder usarlos fuera de este módulo habría que hacer un "casting", puesto que ModuleHandler, que es quien da acceso a todos los módulos al Controller, los almacena como Module. Por esto es mejor que sólo utilices como "public" los que vienen por defecto de Module.

2. ModuleX.cpp: Implementa el proceso de detección de un programa de ocultación de contenidos sobre unos determinados medios.

Normas a seguir:

- Incluye los siguientes encabezados: #include "ModuleX.h" (definición de la clase), y #include "YMedia.h" (definición del tipo de Medio sobre el que vas a realizar la detección: En el directorio "parser/" puedes encontrar todos las clases que heredan de Media).
- Hay que definir una inicialización para el atributo estático que se definió en ModuleX.h. Por ejemplo: ModuleX ModuleX::theModule = ModuleX(); Así, llamanos al constructor de la clase, que, como ya veremos, se encargará de registrar este módulo en el ModuleHandler.
- En el constructor:
    - Llama al propio de la superclase, con un parámetro: el nombre que definiste en Module.h para identificar unívocamente a este módulo. Por ejemplo: ModuleX::ModuleX() : Module(MODULEX).
    - Indica sobre qué medios es adecuado utilizar este módulo. Típicamente: this->applyTo->push_back(Y), donde Y es un identificador del tipo de Medio, que puedes encontrar en "/Parser/MediaType.h".
    - Inicializa el atributro donde vas a guardar el resultado: this->result = new ResultX();
    - Registra el módulo en el ModuleHandler: this->id = ModuleHandler::get_instance()->register_module(* this);

En this->id guardas un ticket único que devuelve ModuleHandler para cada módulo registrado, que coincide con el número de módulos registrados hasta el momento, aunque carece de utilidad por ahora.

- En el método de detección (apply):
    - Comprueba que el Media que te pasan por parámetro se trata realmente de la clase para la que el módulo esta capacitado detectar: if (find(this->applyTo->begin(), this->applyTo->end(), media->get_id()) == this->applyTo->end()) { throw new UTCException("This module can't handle this type of media");}.
    - Resetea los resultados que se han obtenido anteriormente al ejecutar este módulo: this->result->initialize();
    - Haz el casting apropiado al Media que te llega por parámetro, para poder usar los métodos que provee: YMedia* y = (Y*) media; En el directorio "parser/" puedes encontrar todas las clases que heredan de Media.
    - Maneja las excepciones que se puedan producir en la ejecución del método mediante instrucciones try/catch.

Algunos consejos:

- Incluye #include "UTCException.h" para lanzar excepciones en caso de que haya algún error. Así desde fuera serán capaces de manejar adecuadamente estas circunstancias anómalas.
- Antes de implementar el método de detección (apply), documenta qué es lo que vas a hacer: la estrategia que vas a seguir, la firma a buscar, enlaces a información sobre el programa que vas a detectar…
- En el método "toString", que se encarga de construir un mensaje de información sobre el módulo, sé escueto pero conciso sobre la versión del programa que detectas.
- En el destructor, libera al menos la memoria ocupada por ResultX. En la versión para Windows pueden producirse algunos errores en la ejecución, pero en Linux tendrás problemas si no lo haces.

3. ResultX.h: Interfaz para los resultados de ModuleX.

   Normas a seguir:

- Incluye los siguientes encabezados: #include "Result.h".
- La nueva clase debe heredar de Result, por lo que debes definir los métodos que eran virtuales puros: públicos: bool is_stego(); y char * to_string().; y protegidos: void initialize(); y double calculate_p_value();

   Algunos consejos:

- Recuerda incluir las directivas de preprocesador ifndef/define/endif, para que no haya problemas de duplicidad de definiciones.
- Si es necesario un destructor, es mejor que lo declares "protected", porque si después haces una subclase de este módulo podrás manejar mejor esta operación. Es posible

que también sea útil por el mismo motivo declarar como “protected” lo que en principio debiera ser “private”.
- Si añades más métodos públicos, para poder usarlos fuera de este módulo habría que hacer un “casting”, puesto que ModuleHandler, que es quien da acceso a todos los módulos al Controller, y por ende a los resultados de éstos, los almacena como Module. Por esto es mejor que sólo utilices como “public” los que vienen por defecto de Result.

4. ResultX.cpp: Guarda los resultados de la detección de ModuleX, y es capaz de devolver un p-valor sobre la probabilidad de que el medio tenga contenidos ocultos. Este valor lo calcula en base a los datos que el Module ha guardado en esta clase.

   Normas a seguir:
- Incluye los siguientes encabezados: #include "ResultX.h" (definición de la clase).
- En el constructor llama al propio de la superclase. Por ejemplo: ResultX:: ResultX() : Result ().

   Algunos consejos:
- Antes de implementar el método de cálculo de p-valor (calculate_p_value), documenta qué es lo que vas a hacer: la estrategia que vas a seguir, los valores que vas a tener en cuenta, cómo los ponderas…
- En el método “toString”, que se encarga de construir un mensaje de información sobre el resultado, sé escueto, pues este mensaje saldrá por pantalla (al menos con DefaultPresentation) cada vez que se muestre el resultado de la detección del módulo.

2.2 Añadir un medio nuevo

Ten en cuenta que posiblemente para extraer los datos del nuevo medio, ya se pueda hacer con la superclase Media. Esta clase lee el archivo y provee métodos para acceder directamente a los bytes/char leídos, así como facilidades para buscar información representada como bytes o hexadecimal, o ir a una posición determinada. La manera en que busca información es para formatos downside-up, es decir, que la información empieza en el byte 0 y termina en el último byte (los .bmp utilizan codificación upside-down). Si esto no es suficiente, o necesitas que se comporte de otra manera, sigue los siguientes pasos:

1. YMedia.h: Define la interfaz del medio.

   Normas a seguir:
- Dependiendo de si el medio es una imagen o texto, incluye los siguientes encabezados: #include "Image.h"; o #include "Text.h", respectivamente. Si es de otra clase, tendrás que seguir estos mismos pasos para crear su nueva superclase.
- El constructor debe recibir por parámetro la ruta donde se encuentra el fichero a leer: YMedia(char* path);

- Si se trata de una imagen y la codificación no es downside-up, tendrás que redefinir los métodos virtuales de la clase Media: int go_to_end(); int go_to_begin(); int go_to_next(); int go_to_next(int pos); int go_to_byte(int byte);
- La nueva clase debe heredar de Image o de Text. Si es de otra clase, tendrás que seguir estos mismos pasos para crear su nueva superclase, o heredar directamente de Media.

Algunos consejos:

- Recuerda incluir las directivas de preprocesador ifndef/define/endif, para que no haya problemas de duplicidad de definiciones.
- Si es necesario un destructor, es mejor que lo declares "protected", porque si después haces una subclase de este módulo podrás manejar mejor esta operación. Es posible que también sea útil por el mismo motivo declarar como "protected" lo que en principio debiera ser "private".
- Si añades más métodos públicos, para poder usarlos fuera de este módulo habría que hacer un "casting", puesto que Reader, que es quien crea todos los medios para el Controller, los devuelve como Media. Esto no suele ser un problema, puesto que los módulos que utilicen este nuevo medio, hacen un "casting" a la clase correcta, pues se aseguran de que el medio que reciben es del tipo correcto.
- Si necesitas usar alguna librería externa para manejar este nuevo tipo de ficheros, documenta su uso, dónde se puede encontrar información adicional… Mira en la sección "Añadir una nueva librería" para ver cómo deberías hacerlo.

2. YMedia.cpp: Implementa el acceso a las características del medio.

Normas a seguir:

- Incluye los siguientes encabezados: #include "YMedia.h" (definición de la clase).
- En el constructor:
    - Llama al propio de la superclase, con un parámetro: la ruta que recibes por parámetro.
    - Indica sobre qué medio actuará esta clase. Típicamente: this->id = Y; donde Y es un identificador del tipo de Medio, que puedes encontrar en "/Parser/MediaType.h".
    - En caso de que necesites leer tú mismo el fichero de alguna manera especial (recuerda que esto lo hacía la clase Media), hazlo aquí, y documenta de qué manera lo haces.

Algunos consejos:

- Incluye #include "UTCException.h" para lanzar excepciones en caso de que haya algún error. Así desde fuera serán capaces de manejar adecuadamente estas circunstancias anómalas.
- Igual necesitas usar algunos métodos de transformación o de algún otro tipo que ya se encuentren implementados en el Toolkit. Puedes comprobarlo en "/Modules/Toolkit.h"

3. MediaTypes.h: Indica los distintos formatos de archivos sobre los que los módulos pueden actuar.

   Normas a seguir:

- Si el nuevo YMedia se basa en un nuevo tipo de fichero, añádelo a la enumeración: enum MediaType {JPEG, GIF, BMP, PLAIN_TEXT, Y};

   Algunos consejos:

- Para que los módulos se identifiquen como capaces de manejar este nuevo tipo de formato, deben hacer en el constructor: this->applyTo->push_back(Y), donde Y es el identificador del tipo de Medio que has definido en MediaType.h.

2.3 Añadir una librería nueva

En el directorio "lib/" se encuentran todas las librerías externas. Por sencillez en la instalación de la aplicación, sólo hemos usado librerías estáticas.
Ten en cuenta que debe haber versiones para Windows y para Linux. Crea los directorios "win32" y "Linux" dentro de "lib/libZ/" y en ellos los archivos .a. Para Windows hemos usado el compilador MinGW, y para Linux G++.

2.4 Añadir una utilidad nueva

En "modules/Toolkit.h" están definidas varias herramientas o utilidades que, sin ser propias de una clase, pueden proporcionan ayuda en la realización de funciones varias. Fíjate que todas son "static": así no hace falta instanciar Toolkit para poder hacer uso de ellas. Documenta bien cuál es el cometido de la nueva función, cómo funciona y los parámetros de entrada y salida. Para poder utilizarlas desde cualquier parte de la aplicación, tan sólo incluye #include "/Modules/Toolkit.h" y añade Toolkit:: antes de la llamada (pues es un método estático).

2.5 Añadir una GUI nueva

La GUI por defecto es DefaultPresentation, que está basada en texto presentado por la consola. Para usar otra, hay que añadirla al directorio "presentation/", y además:

1. ZPresentation.h: Define la interfaz de la GUI.

   Normas a seguir:

- Incluye los siguientes encabezados: #include "Presentation.h".
- La nueva clase debe heredar de Presentation, por lo que debes definir los métodos que eran virtuales puros: void show(Result* result), que muestra el Resultado de un módulo; void show(FinalResult* result), que muestra el resultado final para un archivo; void show_error(char* message), que presenta un error; void show_message(char* message), que presenta un mensaje de aviso o de otra índole.

   Algunos consejos:

- Recuerda incluir las directivas de preprocesador ifndef/define/endif, para que no haya problemas de duplicidad de definiciones.
- Si es necesario un destructor, es mejor que lo declares “protected”, porque si después haces una subclase de este módulo podrás manejar mejor esta operación. Es posible que también sea útil por el mismo motivo declarar como “protected” lo que en principio debiera ser “private”.
- Si añades más métodos públicos, para poder usarlos fuera de este módulo habría que hacer un “casting”, puesto que Controller, lo guarda como Presentation. Por esto es mejor que sólo utilices como “public” los que vienen por defecto de Module.

2. ZPresentation.cpp: Implementa la GUI.

   Normas a seguir:

- Incluye los siguientes encabezados: #include " ZPresentation.h" (definición de la clase).

   Algunos consejos:

- Incluye #include "UTCException.h" para lanzar excepciones en caso de que haya algún error. Así desde fuera serán capaces de manejar adecuadamente estas circunstancias anómalas.
- Los métodos void show_error(char* message) y void show_message(char* message) podrían mostrarse usando algún tipo de ventana modal.
- Los métodos void show(Result* result) y void show(FinalResult* result) podrían mostrarse usando algún tipo de TextArea de sólo lectura.
- Dado que esta aplicación es multiplataforma, tendrás que hacer 2 versiones de la interfaz de usuario: una para Windows y otra para Linux. Probablemente sea más adecuado usar algún tipo de herramienta/lenguaje que te permita usar el mismo código para cualquier plataforma, como WxWindows (WxWidgets).

SECCIÓN III:

UTILIDADES

Lista de Contenidos

1. StegDetect

1.1 Introducción

StegDetect es una herramienta automatizada de detección esteganográfica de contenidos en imágenes, es capaz de detectar diferentes métodos esteganográficos que embeben información en imágenes JPEG.Actualmente los métodos que detecta son:

-Jsteg

-JpHide

-Invisible Secrets

-Outguess 01.3b

-F5

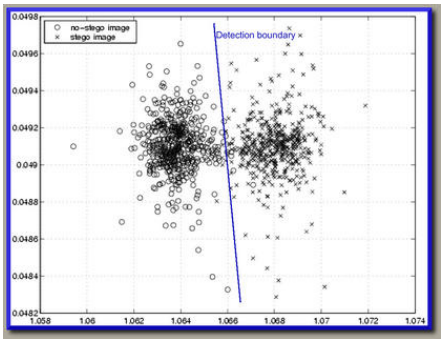
-AppendX and Camouflage

StegBreak se usa para lanzar ataques de diccionario contra Jsteg,JPHide y Outguess,ambos softwares han sido desarrollados por Niels Provos.

StegDetect 0.6 funciona haciendo un "análisis discriminatorio lineal".Dado un número de imágenes normales y un número de imágenes con contenidos ocultos por una nueva aplicación esteganográfica,StegeDetect puede automaticamente determinar una función de detección lineal que puede ser aplicada a un número ilimitado de imágenes.

El análisis discriminatorio lineal realiza los cálculos devolviendo un hiperplano dividido que separa las imágenes que no tienen estegos de las imágenes que sí.El hiperplano es caracterizado por una función lineal.

La función aprendida puede ser salvada para usarla posteiormente con nuevas imágenes.



1.2 Adaptación en UTC

Uno de los objetivos principales de UTC era que incluyese Stegdetect 0.5, es decir, que detecte al menos los programas detectados por la herramienta de Provos. Esto llevó a incluir en nuestra herramienta los módulos de detección para JSteg, JPHide y Outguess, y para ello existían 3 opciones:

- Traducir nosotros mismos las ideas de Provos, sobre la detección de estos módulos, a código fuente;

- Modificar el código fuente de Stegdetect para adaptarlo a la estructura de UTC;

- Por último, hacer una llamada al programa ejecutable de Stegdetect cuando tuviéramos que detectar contenidos ocultos con JSteg, JPHide o Outguess;

Gracias a la licencia GPL a la que está sujeta Stegdetect, decidimos que lo más práctico sería la segunda opción. Permitiría usar un método de detección ya desarrollado y probado, sin vulnerar derechos de autor (ya que se permite la copia

y modificación de su código siempre que se respete la autoría original), y evitaría volver a desarrollar algo que ya existe.

Una vez decidida la forma de implementar los 3 módulos, el siguiente paso era estudiar el código fuente de Stegdetect. El primer problema que localizamos fue la codificación en ANSI C y la dependencia del sistema operativo, ya que está pensado para funcionar bajo UNIX. UTC está escrito en C++, y es multiplataforma. Por lo tanto, había que cuidar las definiciones de tipos: algunos que se incluyen por defecto en plataformas UNIX no existen en las cabeceras de el entorno de programación (Borland CBuilderX). Asimismo, otros tipos predefinidos tienen distintos tipos base en Windows y Linux, por lo que precisaríamos compatibilidad entre ambos.

Tras resolver estos inconvenientes, pasamos a trasladar código original a módulos de UTC. Stegdetect no tiene estructura modular: concentra los métodos de apertura de ficheros y los métodos de detección principales en el mismo archivo (stegdetect.c). Desde aquí se realizan las llamadas al resto de archivos, uno de los cuales (common.c) incluye los métodos específicos de detección de marcas y aquellos métodos de tratamiento de imágenes JPEG. Otros archivos incluidos en Stegdetect proporcionan formas de tratamiento de ficheros, extracción de contraseñas y contenidos ocultos, tablas para funciones estadísticas, técnicas de cifrado, … funcionalidad que no resultaba útil para nosotros.

De hecho, casi todo el código se extrajo de common.c y stegdetect.c. El contenido de common.c se repartió entre JPEGMedia (métodos de tratamiento de archivos JPEG) y los módulos particulares de JPHide, JSteg y Outguess. El archivo principal de Stegdetect se repartió entre el archivo Toolkit y los módulos particulares. De estos contenidos, se ha modificado:

- Las salidas estándar (printf) se han eliminado;
- Los valores que se calculan al aplicar un módulo sobre una imagen, se guardan en los correspondientes objetos Result, para facilitar el cálculo del p_valor según criterio del usuario. Si en un futuro se quieren ajustar los parámetros de decisión del p_valor, no hay más que modificar el método correspondiente en las clases Result;
- Se han añadido conversiones explícitas de tipos, y eliminado tratamientos de error y demás código inútil en UTC;

Una vez compilado el programa (sin errores), procedimos a estudiar su funcionamiento. Stegdetect se apoya en la librería jpeglib (creado por Independent JPEG Group's software, cuya web es www.ijg.org), y la distribuye junto con su código fuente. Pero resultó no ser la librería estándar, sino una modificación propia de Provos en la que añade funcionalidad para apoyar su herramienta. Su descripción y proceso de adaptación se incluyen en el siguiente apartado.

Finalmente, tras la obtención de una librería jpeglib (o libjpeg, los autores utilizan ambas terminologías) adecuada para UTC, se procedió a la corrección concreta de cada módulo. JPHide no provocó grandes problemas; JSteg precisó de una depuración en profundidad con buenos resultados, mientras que Outguess aún provoca algunos problemas. Ante la imposibilidad de depurarlos, nos pusimos en contacto con Provos para solicitar ayuda, que declinó amablemente.

2. LibJPeg

2.1 Introducción

Como hemos comentado anteriormente, Stegdetect utiliza la funcionalidad proporcionada por esta librería para gestionar los ficheros de tipo JPEG.
Este paquete proporciona código C para implementar la compresión y descompresión de imágenes JPEG. JPEG es un método de compresión estandarizado tanto para imágenes en color como para aquellas en escala de grises. JPEG está destinado a comprimir imágenes de la "vida real": dibujos lineales, animados o imágenes no reales no son su punto fuerte. JPEG tiene pérdidas, es decir, la imagen de salida no es idéntica a la de entrada. En cualquier caso, en fotos e imágenes típicas, se consiguen niveles de compresión muy buenos sin apenas pérdida de calidad al ojo humano, y niveles excelentes si la calidad de la imagen de salida no es fundamental.

2.2 Adaptación en UTC

Durante el proceso de prueba del código adaptado de Stegdetect, llegó el momento de compilar y enlazar la librería. En un primer momento, pensamos utilizar la librería proporcionada por Stegdetect: ésta incluía makefiles para casi todas las plataformas y entornos, pero no para CBuilderX. Obtuvimos makefiles para C++Builder 5, y compilamos pero no pudimos enlazar la librería.

El siguiente paso fue descargar los fuentes de libjpeg directamente de la web, suponiendo una posible incompatibilidad en la distribución incluida en Stegdetect. Tras varias pruebas no se pudo compilar. Pasamos a compilar en Linux, pero la librería creada bajo Linux era incompatible con Windows, por diferencias en las cabeceras incluidas. El error residía en que, aún compilando correctamente en Windows, no modificábamos correctamente el archivo jconfig.h, en el que se declaran tipos que luego entran en conflicto con las cabeceras de CBuilderX. Dándonos cuenta de esto, conseguimos compilar en Windows la librería libjpeg descargada de la web.

En este punto, no conocíamos las modificaciones de Provos sobre el código de libjpeg, y dimos por correcta la relación UTC-libjpeg. Todo funcionaba, salvo

JSteg. En nuestra herramienta, cada vez que se calculan los coeficientes DCT en JSteg, éstos eran nulos. Depurar en CBuilderX es complicado, ya que no ofrece utilidades apropiadas en este sentido (CodeGuard, breakpoints inválidos, etc.). Y depurando Stegdetect en Linux, descubrimos que la librería no era igual. Provos incluye unas funciones que denomina "callbacks" para discriminar qué función se utilizará para el cálculo de los DCTs. A partir de aquí, compilamos la librería modificada de Provos en Windows, obteniendo resultados satisfactorios.

Debemos aclarar que la elección de la librería a utilizar se basó en sus posibilidades para obtener datos de archivos JPEG, y no de su inclusión predeterminada en Stegdetect. En un estudio inicial, incluído en este documento, se valoró la funcionalidad proporcionada por las distintas librerías existentes y las necesidades que teníamos, concluyendo con la elección de libjpeg como soporte más adecuado.